SEAdog

Electronic Mail System

Version 4.0

TABLE OF CONTENTS

BLANK PAGE

INTRODUCTION

SEAdog is a full-featured electronic mail system based on the personal computer and using standard telephone lines. It is designed to be easy to use, yet powerful. But above all, it is inexpensive to operate. It can be used to send and receive messages and/or files. Any sort of file can be sent using SEAdog, including programs, word processor files, binary data files, and spreadsheets. A CRC 16 error checking polynomial is used to ensure a near-zero error rate.

SEAdog is a sophisticated store-and-forward mail system which can be configured in a virtually unlimited number of network topologies (more on this later). Unlike some network systems, the end user need never concern himself with network routing -- it all happens automatically. The user just enters and reads messages, the system takes care of the details.

SEAdog can not only send messages and files, it can be configured to allow users to request files or file updates from other SEAdog systems. This can also be automated, thus allowing automatic distribution and updates of any files or programs.

You will need the following hardware in order to use SEAdog:

- An IBM PC, XT, AT, or compatible, with 256k of main memory and a floppy disk drive. A fixed disk is recommended, but not required. Any IBM compatible display may be used.

- A Hayes command set compatible modem.

- A telephone line with access to an outside line, and which can be reached by direct dialing from outside. An internal PBX system can be used, but only if no outside access is required.

- A printer connected as LPT1: can be used to advantage, but is not required.

You do *not* need to dedicate a machine to SEAdog; it operates during your spare time.

SEAdog uses the FidoNet Electronic Mail Protocol, as defined in the document, *A Basic FidoNet Technical Standard*, published by the International FidoNet Association (IFNA). The FidoNet Protocol is a public domain electronic mail standard originally developed by Tom Jennings for the Fido bulletin board system. For more information about the FidoNet Protocol, please write to:

The International FidoNet Association
P.O. Box 41143
St. Louis, Missouri 63141
United States of America

There are several advantages to using the FidoNet Protocol, not the least of which is that a great many utilities and programs are available from many different vendors for doing various things with electronic mail. Please contact IFNA at the above address for more information.

The heart of SEAdog is the network mail server, MAILER.EXE. This is the program that places and receives phone calls, handles message routing, and so forth. It is left running when you would normally turn your machine off.

Most people turn off their machine at the end of the day when they go home. The machine is then idle and doing nothing until they return in the morning. With SEAdog, you instead leave the machine turned on and running the mailer. The mailer waits until the early morning hours (normally 4AM Eastern time, 1AM Pacific time) when phone rates are cheapest, and then begins placing phone calls to other SEAdog systems to pass them your outgoing mail. When you return in the morning, you "take down" (ie. terminate) the mailer and read your mail.

This can be altered in a number of ways. For example, in a commercial environment where you have a great deal of message traffic, you may wish to have a longer mail window than the conventional one hour. Or, if you have a printer, you may wish to have your incoming mail printed out for you each morning. All of this, and more, is possible.

A SEAdog mail system can be configured as a single network or as multiple networks. In a single network, everyone has a distinct *node number* which defines their *network address*. This can be any integer from 1 to 32767, and is different for each machine. You can think of node numbers as being similar to phone numbers.

If you use multiple networks, then every node will also have a *network number*. For example, if you have several nodes in San Francisco and several more in New York, then you could designate San Francisco as network 1, and New York as network 2. Every node in New York would have the same network number, but different node numbers. You can think of network numbers as being like area codes. It doesn't matter if a node in New York has the same node number as a node in San Francisco, as long as they have different net numbers.

We generally specify network addresses as two numbers separated by a slash ("/"). The first number is the *network number*, and the second number is the *node number*. For example, 107/8 would mean node 8 in net 107. If you are using a single net mail system, then you don't need to worry about network numbers at all; just enter net addresses as a single number (the node number).

When you give the address of someone in your own net, you don't have to give the net number (though it doesn't hurt). Again, this is like area codes. You don't have to dial your own area code to call next door.

All of this can be made even simpler by using a *user list*, which is like a phone book. If you have a user list, then SEAdog will look up the name of whoever you are sending your message to and fill in the net address for you. (Don't you wish the phone company did that!) Refer to the **User List** section for details on creating a user list.

If you have a fairly small number of nodes in a fairly localized area, then it is easiest to set up and maintain a single network system. However, if you have a large number of nodes in widely scattered areas, then it is best to have a multiple network system. Refer to the **Node List** section for details on establishing and maintaining a multiple network mail system.

Copy protection

SEAdog is not copy protected in any way. You may make as many copies of the software as you like. However, the software provided on one distribution disk may only be used on one machine at any given time.

When a SEAdog system receives mail, it checks to see if the system sending the mail came from the same distribution disk. If it did, then SEAdog will leave you a message telling you that it has detected a duplicate copy, and which node is running the duplicate. If it keeps receiving mail from duplicate copies, then it will eventually refuse to accept mail from duplicates any more. It will still accept mail from non-duplicate copies.

The point of all this is to make it possible for you to evaluate SEAdog when all you have is one copy. If you don't get enough mail transfers for your evaluation, then simply remove it from your test system and install it again.

OPERATION

SEAdog consists of the following programs:

MAIL.EXE This is the primary user interface program, and is used to enter and read messages.

MAILER.EXE This is the network mail server, and is left running at night to place and receive calls.

SEND.EXE This is used as an easy method to send only a file, with no attached message.

GET.EXE This is used as an easy method to request a file or an update from another SEAdog system.

TELL.EXE This is used as an easy method of sending "canned" messages.

RENUMBER.EXE This is a stand-alone message base renumbering utility. The message base can be renumbered manually from MAIL, but this utility provides a way to automate the process.

ROBOT.EXE This is the automated file mailer, and is used for automatic file distribution and updating.

TWIX.EXE This is used to automate the printing of received messages.

ULMAINT.EXE This is used to simplify the creation and maintenance of a user list.

The ROBOT, TWIX, and RENUMBER programs are used to automate certain common functions. Robot and TWIX are each described in their own sections.

The other programs together comprise a flexible and powerful user interface to the SEAdog mail system. Most of them are described in detail in the **User Interface** section.

The SEAdog MAILER program

The MAILER program is the heart of the SEAdog mail system. The other programs are all, in one way or another, interfaces to the files and data of MAILER. MAILER is the network mail server, and is left running on your machine overnight. This is the program that actually places and receives calls, so you *must* leave it running overnight if you wish to send or receive mail.

The MAILER program is generally invoked from a batch file (typically named "SEADOG.BAT"), so that it can trigger *external events*, such as running Robot or Twix. See the **Installation** section for details about this.

Most of what MAILER does is controlled by its configuration file, CONFIG.DOG, which will be described in detail in the **Installation** section. It normally runs as a "hands-off" program, but there are a few things you can do by entering commands at the keyboard.

1) **Take it down**; This is done by entering a Control C (pressing the "C" key while holding down the "Ctrl" key) or a Control Break. The mailer will finish up anything it was doing, including ending the current mail event (if any), and return control of your system to you.

2) **Invoke the user interface**; This is done by entering an Alt M (pressing the "M" [for "Mail"] key while holding down the "Alt" key). The mailer will finish up anything it was doing and invoke the MAIL program. You can then enter and read messages. When you exit the mail program, the mailer will resume what it was doing before.

3) **Force a call**; This is done by pressing the "C" (for "Call") key. If the mailer was waiting to make a call to another system, then it will end the wait and call immediately.

4) **Abort a call**; This is done by pressing any key *after* it has dialed, and *before* it connects with the other system.

5) **Abort a file transfer**; This is done by pressing the "Esc" key while the file is being transferred.

6) **Get a status report**; If the mailer is waiting to make or to get a call, then entering a question mark (pressing the "?" key) causes it to print a status report on the mail it has sent or received so far.

7) **Give a DOS command**; This only works when the mailer reports that it is waiting for a call or event. This is mainly intended to allow you to use the GET and SEND commands while the mailer is running, but you can use it for almost anything.

8) **End a mail event**; This is done by entering an Alt Q (pressing the "Q" key while holding down the "Alt" key) when the mailer is waiting to make or to get a call. The current mail event is ended immediately, and the mailer returns to "waiting for call or event."

9) **Restart an event**; This is done by entering an Alt R (pressing the "R" key while holding down the "Alt" key) when the mailer is waiting for a call or an event. any mail events that would be active but have been ended early (either by being dynamic events, or by the use of "Alt R") will be restarted.

10) **Exit with an error level**; This is done by pressing one of the function keys. When you press a function key, MAILER will exit with an error level equal to ten times the key number (F1=10, F2=20, and so on). This is used to provide local console access for certain bulletin board systems, and to allow external events to be manually triggered. Most users of SEAdog will have no use for it.

If all this seems a bit confusing at first, don't worry about it. All you really need to know to start is how to take down the mailer so you can have your machine back, and that's easy. Just enter a Control C or a Break.

you may also enter a config file to use

How MAILER operates can be modified slightly by use of command line switches. The following command line switches are currently supported:

/t **Terminate**; This tells MAILER to terminate (return to DOS) as soon as it ends a mail event, or as soon as it finishes taking a call. This is intended mainly for use by the "NOW" options of SEND and GET. Please refer to the **Crash Mail** section for more details about crash priority mail, and to the **User Interface** section for more information about SEND and GET.

/a **Alert**; This tells MAILER to alert you by sounding a tone whenever it receives mail or ends a mail event.

For example, assume you are expecting to receive some daytime crash mail. You want SEAdog to take the call when it come in, but you also want to get on with your work as soon as it is done. Further, you want to be notified when the call is over so that you know your machine is free again. You would type:

```
mailer /t /a
```

You don't *have* to use the /t flag to get your machine away from SEAdog. You can always make SEAdog exit by entering a break.

Remember, you *must* leave MAILER running at night if you are going to send or receive any mail.

INSTALLATION

Getting started

You will need to install SEAdog on your system before you can send mail. There are a great many possible ways that SEAdog can be installed, but the simplest way to begin will be to use the automated installation process. Place the SEAdog distribution disk in your A: drive and type:

```
a:install
```

You will be asked a series of questions about what sort of system you have, and how SEAdog should be installed for you. If you get "stuck" at any point, then take all the time you need to answer correctly. If you wish, you can press *Ctrl Break* to stop the installation process at any point. Don't be afraid of making a mistake and losing everything -- you can always repeat the installation later if it didn't come out right the first time.

Floppy disk installation

When you install SEAdog on a floppy disk, it will assume that you have two floppy disk drives available. You will need a bootable floppy disk to put the programs on. To create a bootable floppy, put your DOS disk in drive A:, a blank disk in drive B:, and type:

```
a:format b: /s
```

When the disk is finished, put the SEAdog distribution disk in drive A: and type:

```
a:install
```

When it's all done, take the disk out of drive B: and label it as your *SEAdog system disk*. Create a blank, formatted disk and label it as your *SEAdog message disk*. To use SEAdog, you should put the SEAdog system disk in drive A: and the SEAdog message disk in drive B:, and then reboot your system.

Fixed disk installation

Using SEAdog is a bit simpler when you have a fixed disk. Of course, using almost *anything* is simpler when you have a fixed disk. Just put the SEAdog distribution disk in drive A: and type:

```
a:install
```

Then answer each question as it comes up. The installation process will create the following directories on your disk:

\MAIL	This will hold the various SEAdog programs and files.
\MAIL\MESSAGES	This will hold your messages.
\MAIL\FILES	This is where files mailed to you will go.
\MAIL\HELP	This will hold the various help files.

If it needs to, then the installation process will also create or modify your CONFIG.SYS file to contain the following two statements:

```
files=20
buffers=20
```

Please refer to your DOS manual for more information about the CONFIG.SYS file and what it can contain.

The installation process will also modify your AUTOEXEC.BAT file to contain the following two statements:

```
set SEAdog=C:\MAIL
path C:\MAIL
```

This may not be exactly what it does, as the right thing to do depends on what you already have defined. The SET statement defines an environment variable that tells SEAdog where to find its programs and files. The PATH statement tells DOS to look in the SEAdog

work area when you type a command. These two together make it possible for you to use the various SEAdog programs at any time. Please refer to your DOS manual for more details about the SET and PATH commands.

Last, but not least, the installation process will create a basic CONFIG.DOG file for you. This file tells SEAdog who you are and how to do things for you. You can modify this file using any normal text editor (such as Edlin). We'll describe what can go in it later.

Setting up your modem

You'll need to make sure your modem is set up properly. This usually can't be done by a program, so the automated installation process can't take care of it for you.

Exactly how you set up your modem depends on what brand you have. Most modems set these things by using *DIP switches* (little tiny switches that you set with a paper clip). Sometimes you have to remove a cover plate to get at the switches, sometimes not. Some modems use *non-volatile RAM* (NVRAM), which you set by running a communications program and giving commands to the modem. Check the manual for your modem to see exactly what to do in your case.

You want the switches set for the following:

1) Verbal response codes.
2) Do not answer incoming calls.
3) CD and DTR reflect reality.
4) Extended response set.

SEAdog uses verbal response codes for two reasons. The first is that the verbal response set is nearly universal among modem manufacturers, so SEAdog can use a wide variety of modems. The second is that verbal responses are more convenient for you when you use your modem during the day.

You will note that we are telling you to set your modem so that it doesn't answer the phone. How can this be if your system is supposed to receive mail? SEAdog uses your modem overnight to send and receive

mail, but the modem and the phone line are still available for your use during the day. Human-type callers during the day do not appreciate having a modem answer the phone. For this reason, SEAdog enables autoanswer when it starts up, and disables it when it gives your machine back to you.

SEAdog uses the carrier detect signal to tell when it has a call, and uses the DTR signal to tell the modem to hang up. For this reason, the modem must be set to support them both, and you must have a modem cable which transmits these signals.

Your modem must be *fully cabled*. This means that all of the status signals from the modem are passed to the computer, and all of the control signals from the computer are passed to the modem. In general, if you go to any computer store and tell them what type of modem and computer you have, they will sell you a $50 cable (made from about $5 worth of parts), and it'll work fine.

The extended response set is required if SEAdog is to detect baud rate properly, as it is taking the modem's word for it. Specifically, SEAdog will recognize any of the following responses:

```
CONNECT
CONNECT 1200
CONNECT 1275
CONNECT 2400
CONNECT 4800
CONNECT 9600
CONNECT FAST
CONNECT 19.2
CONNECT 19.2K
CONNECT 19200
SET TO 300
SET TO 1200
```

It's okay if your modem adds anything to the *end* of any of these responses. For example, if your modem sometimes says something like "CONNECT 2400 RELIABLE", then that won't be a problem.

When SEAdog is attempting to dial another system it will recognize (though it does not depend on) any of the following:

```
NO CARRIER
BUSY
VOICE
NO TONE
NO DIALTONE
NO DIAL TONE
RRING
RINGING
```

If your modem can give any of the above status messages, then you will probably want to enable them.

Switch settings for the Hayes Smartmodem 1200

You will need to remove the front cover from the Hayes modem in order to reach the configuration switches. *Gently* pry up the tabs on either side of the front cover and slide it off. If you are not sufficiently gentle, you will break a tab. Set the switches as follows:

1.	UP	Program supports DTR.
2.	UP	Result codes in English.
3.	DOWN	Send result codes.
4.	n/a	Echo commands.
5.	DOWN	Do not answer incoming calls.
6.	UP	Enable carrier detect.
7.	n/a	Single or multiple phone jack.
8.	DOWN	Enable command recognition.

Your CONFIG.DOG file should contain the statement:

```
Modem H12
```

Switch settings for the USR Courier 2400

The configuration switches on the USR Courier 2400 are reachable through a hole in the bottom of the case. Turn the modem over and set the switches as follows:

1.	OFF	Program supports DTR.
2.	OFF	Result codes in English.
3.	ON	Send result codes.
4.	n/a	Echo commands.
5.	ON	Do not answer incoming calls.
6.	OFF	Enable carrier detect.
7.	n/a	Single or multiple phone jack.
8.	ON	Enable command recognition.
9.	n/a	Escape code operations.
10.	n/a	Reserved for future use.
11.	OFF	Normal pin assignments.

Your CONFIG.DOG file should contain the statement:

```
Modem Courier
```

Switch settings for the Novation SmartCat Plus

The configuration switches on the Novation SmartCat Plus are reachable through a hole in the bottom of the case. Turn the modem over and set the switches as follows:

1. OFF Enable command recognition.
2. ON Result codes in English.
3. OFF Send result codes.
4. n/a Echo commands.
5. OFF Do not answer incoming calls.
6. OFF Answer at 300 or 1200 baud.
7. ON Use full response set.
8. ON Enable carrier detect.

Your CONFIG.DOG file should contain the statement:

```
Modem H12plus
```

Switch settings for the Hayes 2400

Setting the switches on the Hayes 2400 baud modem is a snap. It has no switches. Instead, your CONFIG.DOG file should contain the following statement:

```
Modem H24
```

Switch settings for the Everex EV-920

The Everex EV-920 (also known as the Evercom II) also has no switches. However, it has two jumpers which indicate whether it is COM1, COM2, COM3, or COM4. SEAdog can only use the EV-920 if it is configured as COM1 or COM2.

To use the EV-920, your CONFIG.DOG file should contain the following statement:

```
Modem H12plus
```

Switch settings for the MultiTech MultiModem 224E

The configuration switches on the MultiTech MultiModem 224E are reachable through a hole in the bottom of the case. Turn the modem over and set the switches as follows:

1. UP DTR normal.
2. UP Verbal (word) responses.
3. DOWN Enable command responses.
4. n/a Enable command character echo.
5. DOWN Disable auto-answer.
6. UP Carrier detect & data set ready normal.
7. n/a Single or multiple phone jack.
8. DOWN Enable command mode.

Your CONFIG.DOG file should contain the statements:

```
Modem setup AT &F B1 &E1 M0 $Q0 X1 S0=1 S10=20
Modem baud 2400
```

Configuring SEAdog

SEAdog uses a file named CONFIG.DOG to tell it things about how you want it to work. The automated installation process creates a basic configuration file that is probably what you want, but it's easy enough to change it.

The SEAdog configuration file is a standard text file, named CONFIG.DOG, which is located in your SEAdog mail directory. This file can be created and maintained using any standard text editor (such as EDLIN).

Each line of the configuration file is either blank, or begins with a *command verb*, usually followed by arguments. The words on each line are separated by spaces. A semicolon starts a comment, and everything from the semicolon to the end of the line is ignored.

All of this is a lot easier than it sounds. Here's an example of something that might be found in a CONFIG.DOG file:

```
name Thom Henderson ;my name
node 13/1           ;my network address
mail C:\MAIL\MSGS   ;where to keep my messages
files C:\MAIL\FILES ;where to put received files
```

That doesn't look so awful, now does it? We'll go through each of the configuration verbs one by one, and then look at a more involved example.

NAME <text>
USER <text>
ADMIN <text>

These tell SEAdog your name. If you leave this out, then the user interface will ask you for your name each time it is run. You can also have more than one NAME statement, in which case the user interface will print the names it knows about, and ask you which one you are.

If you use USER instead of NAME, then this tells SEAdog that it should restrict the range of choices available to you. For example, a USER cannot request return receipts or audit trails.

If you use ADMIN instead of NAME, then this tells SEAdog that you are an administrative user. ADMIN is mainly useful if your node is operating as a routing node within your network. Only an ADMIN may request an audit trail.

If your node has multiple users, then you can mix NAME, USER, and ADMIN statements. SEAdog will keep track of who is what.

NET <net#>
NODE <node#>
NODE <net#>/<node#>

These tell SEAdog what your own network address is. If you are on a single net system, then you would just use the NODE verb followed by your own node number. If you are on a multinet system, then you could use either separate NET and NODE statements, or just combine them both into one NODE statement. For example:

```
net 107
node 7
```

means the same thing as:

```
node 107/7
```

Whichever form you use is strictly a matter of personal taste.

NODE <zone#>:<net#>/<node#>

This is identical to the NODE statement described above, except that using this format also defines what zone you are in for international addressing.

International addressing by means of zones requires special support software. Please refer to the **Extended Addressing** section for more information about zones and international addressing.

AKA <node#>
AKA <net#>/<node#>

This tells SEAdog that you have an alternate network address. This is primarily used for administrative purposes on the public amateur network. Commercial users will usually not need this.

You can have as many AKA statements as you like, and each one may contain as many alternate network addresses as you can fit on the line. An alternate address may not specify a zone.

MAIL <directory>

This tells SEAdog the name of the directory where you want to store your messages. If you don't specify a separate mail directory, then the message files will be stored in the same directory as the SEAdog system programs and files.

FILES <directory>

This tells the SEAdog mailer where to place files which it receives from other SEAdog systems. If you don't specify a separate directory for received files, then they will be placed in the same directory as the SEAdog system programs and files, which is probably not what you want.

PICKUP <directory>

This tells SEAdog that the files in the named directory are available for other SEAdog systems to request. Any and all files in the named directory are available for pickup. You may specify as many pickup directories as you wish. If you do not specify any pickup directories, then no other system may request any files or updates from your system.

WARNING: Files in a pickup directory are, by definition, *not* secure. *Do not* place any files in a pickup directory if they are considered confidential. This is *only* true of a pickup directory. All other files on your system should be quite safe.

LOG <filename> [<detail>]

This tells the SEAdog mailer to keep a log of its activities, and to write it out to the named file. If the file does not exist, then it is created. The log file is intended mainly for testing and debugging reasons, though many people enjoy perusing it. Unless your company desires logs to be kept for some reason, you don't really need a log file. Besides, it tends to get quite large after awhile.

If you do not specify a log file, then no log is kept. If you don't specify a drive and directory for the file, then it is kept in the same directory as the SEAdog programs and files.

Log entries are displayed on the screen as they are made. All log messages are displayed, even if they are not actually written to the file. Each message begins with a character that indicates what sort of message it is. The following characters are used:

!	This marks a message which indicates a serious, but non-fatal, error condition. Typical examples would be messages reporting the inability to create or delete a file.
*	This marks a message which indicates a general status condition or a change of state. Typical examples are the messages indicating the start and end of mail events or phone calls.
+	This marks a message which reports actions proceeding as expected. Typical examples are messages reporting file attaches, connect messages, and so forth.
-	This marks a message which reports actions proceeding not as expected, but as allowed for. Typical examples are messages reporting busy signals, loss of carrier, and so forth.

(space) This marks a message which is adding detail to an earlier message. Typical examples are the network address report on startup, the report of errors fixed on a file transfer, and so forth.

You can, if you wish, specify what level of detail you want kept in your log file. The following levels of detail are available:

ERRORS This causes only serious error messages to be logged.

TERSE This adds reports of when MAILER starts and stops, and when mail events are begun or ended.

BRIEF This adds the reports made at the end of each mail event, showing what mail packets were sent or received.

NORMAL This adds the reports made at the start of each mail event, showing what mail packets were created and which messages were placed in each packet. This is the default level of log detail which is provided if you don't specify otherwise.

DETAILED This adds the status messages showing each call made or received, and what happened during that call.

VERBOSE This gives the most extensive detail of all, including many messages about file transfer status, line quality, and so forth.

For example, if you wanted SEAdog to record everything possible in a file named SEADOG.LOG, you would use:

```
log SEADOG.LOG verbose
```

The various levels of log detail only control how much detail is recorded in the log file itself. All log messages are displayed on the screen at all times, even if they are more detailed than you want kept in the file.

MODEM <brand>

This tells SEAdog what brand of modem you have. If you have a brand of modem that SEAdog already knows about, then you do not need to use any other modem statements (except possibly MODEM COM2). SEAdog knows about the following types of modems:

H12	Hayes 1200
H24	Hayes 2400
COURIER	US Robotics Courier 2400
H12PLUS	Hayes 1200 compatibles with added features

SEAdog also knows about the old Hayes 300 baud modem, and handles it by default. If you have a Hayes 300, then you don't need any of this.

For example, if you have a Hayes 2400 modem, then your CONFIG.DOG file should contain the statement:

```
MODEM H24
```

Modem type "H12" will generally work with most Hayes compatible modems.

MODEM <port>

This tells the SEAdog mailer which communications port your modem is connected to. Allowable values are "COM1" or "COM2". If your modem is connected to your system as COM2, then your configuration file should contain the statement:

```
Modem COM2
```

If you don't specify what port your modem is connected to, then SEAdog will assume that it is connected to COM1 (which it usually is).

MODEM BAUD <rate>

This tells SEAdog the maximum baud rate which your modem will support. This is so it doesn't try making a call at 2400 baud when you have a 1200 baud modem.

If your modem fits into one of the categories which SEAdog already knows about (see above), then you do not need to have a separate MODEM BAUD statement.

MODEM SETUP <text>
MODEM RESET <text>

These statements tell the SEAdog mailer what command strings it must give your modem when it starts up and when it finishes. If your modem fits into one of the categories which SEAdog already knows about (see above), then you do not need to have separate MODEM SETUP and MODEM RESET statements.

A modem command can contain a vertical bar ("|"), which is used to break a command line into two or more separate commands. For example, if you need to send an "ATZ" command to your modem before sending the rest of the initializer, you could use:

```
modem setup ATZ|AT M0 V1 F1 Q0 X1 S0=1
```

Each time SEAdog gives a command to the modem it waits up to six seconds for the modem to report "OK". If the modem does *not* report "OK", or if it reports "ERROR", then SEAdog will spend up to one minute trying to get the modem to accept the command. SEAdog will log an error message on each failed attempt.

If SEAdog finally succeeds in making the modem submit, it will report that the modem is functioning properly again. Otherwise, it will report that you have a serious modem problem. A modem failure or a modem error is not harmful unless SEAdog is unable to force the modem into submission.

PRINTER SETUP <text>
PRINTER RESET <text>

These statements tell SEAdog what command strings it must give your printer before and after it uses it. The printer setup string will be sent to your printer before any messages are printed, and the printer reset string will be sent to your printer after all messages are printed. You may want to do this if you need to set certain margins or fonts. Please refer to the manual for your printer for more information about what control codes it takes.

Either of these strings may contain an arbitrary character value. Specify the value by putting an ampersand followed by the decimal value in the printer control string.

For example, if you wanted SEAdog to send and escape followed by a letter "B" to your printer before it starts printing anything, your CONFIG.DOG file should contain the statement:

```
printer setup &27B
```

Most likely you will not need to perform any special printer setup.

DIAL <area> [<prefix>][/<suffix>]

This is used to tell your SEAdog how to dial a phone number. The first argument says what phone numbers are to be edited, by saying what they begin with. The second argument says what to use instead. In the usual case, all that is necessary is to tell it not to dial your own area code. For example, if you were in area code 201, you would say:

```
DIAL 1-201-
```

This tells SEAdog to strip the "1-201-" from the beginning of any phone number that has it. (This assumes that all phone numbers are stored in the

form "1-aaa-xxx-nnnn".) Or, if you had to dial 9 and then pause to get an outside line, you would say:

```
DIAL 1-  9.1-
```

(The period gets translated into a pause.) If you have to dial an account code (or something) after the number, you could say:

```
DIAL 1-201-  9/.1234
```

This tells SEAdog that the number "1-201-694-3348" should be dialed as "9694-3348,1234".

DIAL <method>

This is used to indicate what sort of dialing method to use when dialing the phone. The possible methods are:

TONE This indicates that touch-tone dialing may be used.

PULSE This indicates that your local telephone service does not include touch-tone dialing, so pulse dialing must be used.

If you do not specify otherwise, then touch-tone dialing will be used.

EVENT <tag> <day> <start> [<stop> [<modifier>]]

This defines an event. There are two main types of events: mail events, and external events.

A mail event is a time period during which the SEAdog mailer is to place calls to other systems. The time period is defined by a day of the week (given by its three letter abbreviation), or "ALL" (meaning every day), plus a starting time and an ending time. Times are given in the form HH:MM on a twenty-four hour clock. In other words, 5:00 means 5 AM, and 17:00 means 5 PM.

If the ending time is not given, then it defaults to one hour after the starting time.

If the ending time is earlier than the starting time, then it means that time the next day. For example:

```
Event A Mon 23:00 01:00
```

defines event A as stretching from eleven PM Monday to one AM Tuesday.

Mail events are *tagged* with a single letter between A and W (uppercase and lowercase are equivalent). Mail event tags are used to indicate what sort of mail routing should be done. See the **Routing Tags** section for details of mail event routing.

An example of a typical mail event definition would be:

```
Event A All 4:00 5:00
```

This defines a mail event with the tag "A", which happens every morning from 4:00 AM to 5:00 AM.

A mail event can have one or more *modifiers*. These change how SEAdog will act during the mail event. The following modifiers are available:

CRASH This turns the event into a *crash mail event*. This means that only crash priority mail will be sent during this event. Please refer to the **Crash Mail** section for more details.

DYNAMIC This turns the event into a *dynamic event*. This means that the mail event will not run past the ending time, but will end sooner if no more calls will be placed.

BBS This enables bulletin board access during the mail event. You will not be able to delete messages which are being held in mail packets. You should avoid renumbering the network mail area from the bulletin board, or the results will be predictable, and very bad. The vast majority of SEAdog installations will have no use for this modifier.

An example of a typical mail event definition using modifiers would be:

```
Event S All 4:00 5:00 Crash Dynamic
```

This defines a dynamic crash mail event with the tag "S", which happens every morning from 4:00 AM to 5:00 AM, or ends sooner if all mail is sent.

An external event signals some task to be performed external to the SEAdog mailer. This could be running Robot shortly before any mail events, or running TWIX in the morning, or anything you like.

An external event is defined much the same way a mail event is, with two exceptions:

1) The tag for an external event is "X", followed by a number. For example, "X5" means "external event number 5".

2) External events are by nature dynamic. Adding a DYNAMIC modifier to an external event is legal, but it doesn't get you anything.

The number following the X is actually an *error level number*. The mailer invokes an external event by terminating and passing the error level to the operating system. The batch file used to run the mailer can then intercept that number, take the appropriate action, and re-invoke the mailer. An example of this is given later on.

The CRASH and BBS modifiers may be used on external events to modify how they work, though few SEAdog installations will ever have need of them.

A *crash external* event is one that *only* happens if crash mail is received and unpacked during its window.

A *BBS external* event is just like a normal external event, except that it is ignored when calculating the "time to next event" for use in the "*t" parameter of the BBS command.

EVENT BASE <time>

This defines a "base time" for all of the event statements which follow, up to (but not including) the next EVENT BASE statement, or the end of the file. An event base is a time reference that the event definitions are defined relative to. For example, if your CONFIG.DOG file contained:

```
Event base 4:00
Event A All 0:00 1:00
```

This defines event A as stretching from the base time to one hour after the base time, or four AM to five AM. The event times can also be negative, in which case they refer to times before the base time. For example:

```
Event base 4:00
Event H All -1:00 -0:30
Event G All -0:30  0:00
Event A All  0:00  1:00
```

This defines event H as stretching from three AM to three thirty AM, event G from three thirty AM to four AM, and event A from four AM to five AM.

Event bases are mainly useful for compensating for different time zones, and especially for adjusting to and from daylight savings time.

The default event base is midnight, which makes event definitions relative to the real time. You don't need to use an event base if you don't want to.

MSGBITS [PRIVATE|KILL/SENT]

This tells the user interface what message attributes you normally want when you enter a message. These attributes can still be turned on and off by using your function keys when you enter a message. This just tells SEAdog that you want to start out with them *on* instead of *off*. There are two message attributes that you can set by default. They are:

PRIVATE This tells SEAdog that you want your outgoing mail marked as private, unless you say otherwise

KILL/SENT This tells SEAdog that you want your outgoing messages to be deleted after they are sent, unless you say otherwise.

For example, if your configuration file contained:

```
MSGBITS KILL/SENT
```

then whenever you entered a message, it would by default be deleted after it is sent, unless you say otherwise.

Default message bits do not apply to alternate message areas. See the description of MAIL program utility functions in the **User Interface** section for more about alternate message areas.

KILL <thing>

This is used to tell your SEAdog that certain messages should be killed automatically. There are two "things" that can be automatically killed. They are:

NULL This tells the SEAdog mailer to kill any received message which is null (contains no text). Messages created by SEND, GET, and ROBOT have no text, so you may want to use this to keep from seeing those messages. ← ArcMail !

RCVD This tells the SEAdog user interface to kill any message once it has been received (read by its recipient). You will probably not want to use this.

HELP <directory>

This tells SEAdog where all of its help files are stored. If this is not given, then SEAdog assumes its help files are all in the SEAdog work area.

RETRY <with> [<without> [<minwait>]]

This tells the SEAdog mailer how many times to call another node during a mail event. If the mailer reaches the other node and successfully gives it the mail, then it won't call back. But if it can't get through, then it will try again. This tells it how many times to try.

Normally, one SEAdog will call another up to thirty times in a given event, unless it gets a carrier. If it gets a carrier signal but cannot establish communications with the other SEAdog, then it assumes that the modem is turned on, but SEAdog is not running. After the third call when a carrier is detected, SEAdog will not try again for the remainder of the mail event.

The first number after the RETRY verb is the number of times to call when a carrier is detected (normally 3). The second number, if given, is the number of times to call when a carrier is *not* detected (normally 30).

The third argument, if given, defines a minimum number of seconds to wait between phone calls. This is provided for use in some countries where it is not legal to place an outgoing call within a certain period of placing or receiving another call. Check with your country's telephone company to find out if this applies to you.

If any given connection calls for the use of a script, then it will not count as a connection with carrier unless the script is fully completed. See the **Scripting** section for more information about using scripts.

You should not need to use this statement unless you are having severe difficulties with your phone lines, or severe legal restrictions on your phone usage.

BANNER <text>

This defines a brief message which is to be sent over the phone whenever a call is received. The typical SEAdog system should only be called by other SEAdog systems, so this won't ordinarily be needed. The default banner is "Private mail system -- Please hang up".

BBS <command>

This specifies a command to be used to invoke a BBS program. When a call is received SEAdog will wait at least five seconds for a TSYNC character. If it does not receive a TSYNC in that time, then it will normally log the user off. However, if the BBS verb has been given in the config file then SEAdog will invoke another generation of DOS, passing it the command.

The command used to invoke the bulletin board may have any or all of the following in it:

***b** This is replaced with the baud rate at which the connection was established.

***p** This is replaced with a "1" if the mailer is using COM1:, or a "2" if COM2: is in use.

***t** This is replaced with the time in minutes until the start of the next scheduled event. Events which have a BBS modifier are *not* included.

***d** This is not replaced by anything. Instead, it tells the mailer that any dynamic mail events should be reactivated when the bulletin board program is finished. This is only needed if the bulletin board program can create mail which can be sent immediately.

***x** This is not replaced by anything. It indicates that the bulletin board software has loaded SEAdog (instead of the other way around), and that SEAdog should return control to the bulletin board by terminating. The mailer will indicate the baud rate of the caller by terminating with an error level of the baud rate divided by one hundred (3 for 300 baud, 12 for 1200 baud, and so on).

The vast majority of SEAdog systems are not running bulletin board systems, and hence have no use for the BBS statement.

NODELIST <name>

This tells SEAdog where to look for new node lists. Normally, SEAdog looks for the node list in a file named NODELIST.BBS in the SEAdog work area, but you can change that with this statement. For example, if you normally expected your node list to be mailed to you, then you could have a statement like:

```
Nodelist C:\MAIL\FILES\NODELIST.BBS
```

This way, your SEAdog will apply new node lists as soon as possible after they are received.

Sample configuration files

Here is an example of what a CONFIG.DOG might contain:

```
admin   Thom Henderson              ;My name, as a boss
net     13                          ;My net, Mid-Atlantic
node    1                           ;My primary address
aka     13/0 107/8                  ;Other possible addresses
modem   Courier                     ;What type of modem I have
msgbits private kill/sent           ;Normal message flags
mail    C:\MAIL\MSGS                ;Where my mail goes
files   C:\MAIL\FILES               ;Where received files go
pickup  C:\MAIL\WIDEOPEN            ;File pickup area
kill    null                        ;Kill empty messages

event X1 All 4:00                   ;Run ROBOT each night
event A  All 4:00  5:00             ;National mail hour
event X2 All 8:00                   ;Run TWIX every morning
event S  All 8:00 17:00 Crash       ;Daytime crash mail
```

Here's an example of what a more likely CONFIG.DOG would contain:

```
name    Thom Henderson          ;My name
node    107/8                   ;My network address
modem   H12                     ;What type of modem I have
msgbits private kill/sent       ;Default message flags
mail    C:\MAIL\MSGS            ;Where my mail goes
files   C:\MAIL\FILES           ;Where received files go
kill    null                    ;Kill empty messages

event X1 All 4:00               ;Run ROBOT each night
event A  All 4:00 5:00          ;National mail hour
event X2 All 8:00               ;Print received mail each morning
```

And finally, here is an example of the smallest CONFIG.DOG possible:

```
node    8                       ;My network address
event A  All 4:00 5:00          ;National mail hour
```

In all of the above examples, note that the events are listed in order by their starting times. Where an external event occurs at the start of a mail event, the external event is listed first. It isn't always necessary to do this, but it is good practice, and will help ensure predictable results.

External events

In the first two examples we have two external events, defined thus:

```
event X1 All 4:00          ;Run ROBOT each night
event X2 All 8:00          ;Print received mail each morning
```

These would be implemented by the batch file used to invoke the mailer. A sample of such a batch file would be:

```
Statements                 Comments

echo off                   not really needed
:loop                      see below
cd \mail                   move to the SEAdog area

mailer                     invoke the mailer
if errorlevel 2 goto twix  print mail in the morning
if errorlevel 1 goto robot send regular mailings
goto exit                  exit on Break

:twix
twix                       print any received mail
goto loop                  return to the mailer

:robot
robot                      send any scheduled mail
goto loop                  return to the mailer

:exit                      last statement of file
```

Assuming that the above was stored in a file named SEADOG.BAT, then you would enter the command "SEADOG" to bring up the mailer.

Due to the peculiarities of DOS, the "if errorlevel" statements have to be given in reverse order. That is, the higher error levels must come first. Error levels higher than 255 should not be used.

THE USER INTERFACE

The MAIL program

The MAIL program is the primary user interface to the SEAdog electronic mail system. It is used to enter and read messages, as well as to perform a variety of additional functions.

The MAIL program takes full advantage of the display adapter. If you have a color/graphics adapter, then it will take advantage of it and use different colors to help you find your way around the screen. It works equally well with the monochrome adapter.

MAIL may be invoked in either of two ways:

1) By typing the word **MAIL** at the system prompt.

2) By entering an "Alt M" (pressing the "M" key while holding down the "Alt" key) when MAILER is running.

You should use the second method whenever the mailer is running, such as in a multitasking operating system.

In the lower right corner of the screen will be a block of green text that looks something like this:

```
     SEAdog Mail
Messages  10
Highest   15
Last read 14
```

This is the status area, and it tells you about the messages you currently have on file. The word "**Mail**", above, is the action indicator, and indicates that you are reading your mail. This will change from time to time to reflect the action which you are performing. MAIL always starts out "reading mail". If anything goes wrong, then this will turn into a flashing red "**Error**" sign to alert you to the problem.

The other parts of the status panel have the following meanings:

Messages 10 This means that you currently have ten messages on file.

Highest 15 This means that the last (most recent) message you have on file is message number fifteen.

Last read 14 This means that the last (most recent) message which you have read is message number fourteen.

SEAdog messages are numbered starting at one. The lower the number, the older the message. The message numbers on your system have no connection with the message numbers on any other system.

Whenever a message is added, either from you entering a message or from mail being received, it is assigned the next number in sequence. As messages are deleted, they leave "holes" in the sequence. You should occasionally renumber your messages to get rid of the holes (see below).

Just above the status area is a block of yellow or brown text that will look something like this:

Function keys

F1 Exit mail
F2 Kill
F3 Enter
F4 Reply
F5 Print msg
F6 Edit msg
F7 Forward
F8 Utilities
F9 Search
F10 Select

This is your function menu, which shows what actions your function keys will perform. Function keys do different things in different contexts, so this menu will change to show what they do at any given time. We'll explain these functions in more detail shortly.

Just above the function menu is a line of blue (bright on a monochrome display) text to remind you that on line help is only a keystroke away. Any time you are unsure what to do, just enter an "Alt H" (press the "H" key while holding down the "Alt" key), and MAIL will give you an explanation of what is going on.

The top of the screen is the header area, and is used to provide information about the message currently being displayed, such as who it is from, who it is to, the subject, and so forth.

The bulk of the screen is the text area, and is used to display the text of a message.

Wedged in between the header and the text is one line that sometimes pops up in bright red (bright underlined on a monochrome display). This is used for special messages, usually to warn you about something or to ask you a question about something.

Reading messages

MAIL always starts out "reading messages", and returns to it whenever you finish doing something else. As much as possible of your current message (if any) will be displayed on the screen. If the text won't all fit, then a green arrow will appear near the bottom right corner of the screen. If the arrow is pointing down, then there is more text "below" the bottom of the screen. If the arrow is pointing up, then there is more text "above" the top of the screen. If the arrow is pointing in both ways, then there is more text in both directions.

Imagine the screen as being a window, through which you can see part of the message text. You can move the window up and down by using any of the following keys:

Up arrow	This moves the window up one line.
Down arrow	This moves the window down one line.
PgUp	This moves it up several lines.
PgDn	This moves it down several lines.
Home	This moves it to the start of the text.
End	This moves it to the end of the text.

Flipping through messages

Imagine that your mail is in a notebook, with the older messages to your left, and the newer ones to your right. New messages are added to the end of the book as they are received. The pages are numbered, and a message can be deleted by ripping out its page and throwing it away. At any given time you are looking at one page of this book. There are several ways that you can move around in this book.

The most common method is to flip pages. Do this by using your left and right cursor arrows. Pressing the left arrow flips one page to the left, and shows the next older message. Pressing the right arrow flips one page to the right, and shows the next newer message.

You can also flip pages by pressing the space bar or the "Enter" key, which will continue in whatever direction you were going the last time you used an arrow key.

A less common method is to go straight to a page number. You do this by typing the number of the message you wish to see, and then pressing the "Enter" key. For example, if you wanted to see message number ten, you could type "10" and press "Enter".

You can also go straight to the first (oldest) message by entering "Ctrl Home" (pressing the "Home" key while holding down the "Ctrl" key). Or you can enter "Ctrl End" to go straight to the last (newest) message.

There are some other fancy things you can do. Pressing the "+" key will go to the reply to your current message, if any, while pressing the "-" key does the reverse. Also, pressing the "*" key goes to the first message after the last (newest) message you have read. These options are less often used.

But enough about reading mail. Let's move on to some of the other things you can do.

F1 Exit mail

The "F1" key is used to quit whatever you are doing. If you are reading messages, then it means quit using MAIL and return to your operating system. Anywhere else it means quit doing whatever and go back to reading messages.

F2 Kill

When you are reading messages, the "F2" key is used to delete (or *kill*) a message you no longer want to keep around. You will be asked to confirm that you really meant to delete the message, since once it is killed it is gone forever. Press the "Y" key if you are sure, or the "N" key if it was a mistake.

F3 Enter

Use the "F3" key when you want to enter a message. The first question you will be asked is who the message is to be sent to. Enter the name of the person you want to write to.

If the name you enter isn't listed in your user directory, then you will be asked what node to send the message to. Enter the person's node number. Enter a number sign ("#") if you want a list of nodes and node numbers. You can enter more than one node number, separated by spaces, if you want the message to go to more than one node. You can also enter the word "ALL", which means to send the message to every node. In addition, you can enter the name of a file, where the file contains a list of node numbers to send the message to.

You will then be asked if you want to attach one or more files to the message. If you have a file that you want to send along with the message, then press "Y". Otherwise press "N".

If you are attaching files to the message, then you will be asked for the names of the files to send. You can enter one or more file names, separated by spaces. "Path" names are allowed, as are the "wildcard" characters "*" and "?". You'll get a warning message if the files don't exist. If you are *not* attaching files to the message, then you will be asked what the subject of the message is.

You can then enter the text of your message. Type anything you like. Don't worry about hitting the right margin; MAIL will take care of that for you. You can use your four cursor movement arrows to move around over the text you are entering so you can easily fix any mistakes. The "Ins", "Del", "Tab", and "Backspace" keys all work as you'd expect. You can also use the following:

Home	Move to the beginning of a line.
End	Move to the end of a line.
Ctrl Left	Move one word to the left.
Ctrl Right	Move one word to the right.
PgUp	Move the window up one page.
PgDn	Move the window down one page.
Ctrl End	Delete from the cursor to end of line.
Ctrl Enter	Insert a "hard return" (an end of line which is preserved).

Use your function keys to perform any of the following functions:

F1 Discard	Throw this message away.
F2 Save	Save this message for later mailing.
F3 Ins Line	Insert a blank line at the cursor.
F4 Del Line	Delete the line the cursor is on.
F5 Read text	Read the text of a file into a message.
F6 Crash	Mark the message as "crash priority".
F7 Private	Mark the message as "private".
F8 Kill/Sent	Make the message go away once it is sent.
F9 Receipt	Ask for a return receipt.

When you have finished entering your message, press either "F1" to throw it away (you will be asked to confirm this), or "F2" to save it for later mailing.

Sometimes it will seem as if the message is not saved the same way you entered it. The message may "look different" when you read it later. This is because messages are always reformatted to fit your current margins whenever they are displayed. MAIL preserves your paragraph boundaries, but usually not your line boundaries. A paragraph is defined as either starting with an indented line, or separated from the previous paragraph by a blank line, or both.

The exception to this is the *hard return*. A hard return is always preserved, and is indicated on the screen when you are editing a message by a paragraph

symbol. A hard return is entered as a "Ctrl Enter" (pressing the "enter" key while holding down the "Ctrl" key).

This means that if, for example, you want your message to contain a table, you should end each line of the table by holding down the "Ctrl" key when you press "Enter".

F4 Reply

The "F4" key is used to reply to your current message. It is similar to the "F3" (Enter) command, except that MAIL will get the destination name and address from your current message.

There is another way to reply to a message that can be handy if you want your reply to contain text taken from the message you are replying to. Pressing the ">" key while reading a message acts just like pressing "F4", except that the text of the message you are replying to is inserted into your reply, with each line starting with a ">" (to indicate that you are quoting). You can then proceed to edit the message normally, inserting your comments and deleting text you aren't responding to.

F5 Print message

The "F5" key is used to print your current message on your system printer. Obviously, you must have a printer in order to use this. If the message can't be printed for any reason (such as no printer, printer not ready, out of paper, etc.), then a flashing red **"Error"** sign will pop up near the lower right corner of your screen.

F6 Edit message

The "F6" key is used to edit your current message (the one you are reading). You can only edit a message that you, yourself entered using the "F3" (Enter) or "F4" (Reply) function.

Editing a message is just like entering one in the first place. All the keys work exactly the same as they do when you enter a message. The only difference is "F1" and "F2", and they aren't all that different:

F1 Discard Discard changes and leave message alone.
F2 Save Save changes and alter original message.

You cannot delete a message or create a new message with this function. If you "throw away" your changes, the original message will still be intact, as it was before you started to edit it. If you save your changes, then the original message is replaced with your new, edited version.

F7 Forward

The "F7" key is used to forward a message to someone else. You will be asked for the name of the person to forward the message to. If they are not listed in your user directory, then you will be asked to give a network address. This works exactly the same as for entering a message.

You will then be asked if you want to keep a copy of the message for yourself. Enter "Y" if you do, or "N" if you don't. You will also be asked if the message should be forwarded with crash priority. Refer to the **Crash Mail** section for a discussion of crash priority mail.

Any attached files will also be forwarded, if possible. You'll get a warning if they can't be found.

The copy of the message going to the other person will be marked for deletion once it is sent. Any attached files will *not* be deleted.

F8 Utilities

The "F8" key brings up a menu of several lesser used utility functions. You select among them by pressing a second function key. They include:

F1 Quit This leaves the utility menu and returns to reading messages. Use this if you press "F8" by mistake.

F2 Services This brings up another menu of even less used functions. See below.

F3 Request File This is used to request a file from another node. You will be asked who you want to make the request of, and what file you wish to request. The other node must permit file requests for this to work, and only allowed files may be requested.

F4 Req Update This is similar to a file request, except that a file transfer only takes place if the other node has a newer version of the file than you do.

F5 Renumber This is used to renumber your messages. As time goes by and messages are killed, "holes" appear in the numbering of your messages. Every once in awhile you should renumber your messages to get rid of the holes.

F6 Purge This is used to purge (i.e. delete) old messages. You can purge in any or all (or none) of the following ways:

Purge sent; This deletes all messages which are marked as having been sent (mailed to another system).

Purge received; This deletes all messages which are marked as having been received (read by you).

Purge orphans; This deletes all messages which are marked as orphans. A message is marked as an orphan when the mailer tries to deliver it and discovers that it is addressed to a node which does not exist or which has an unpublished phone number.

Purge by date; This deletes all messages more than a given number of days old, as shown by the date the

message was originally entered. You will be asked to enter the number of days. A message one day old was entered yesterday.

F7 Write message This is used to write a message out to a file in plain text form. You will be asked for the name of the file to write the message to. If the file already exists, you will be asked if you want to append the message to the end of the file, replace the file with the message, or quit. Enter any one of "A", "R", or "Q".

F8 Write text This is similar to F7 (Write message), except that only the text of the message is written to the file. The text can, among other things, be read into a new message by using "F5" when entering the new message, thus allowing you to easily create "form letters".

F9 Change area This is used to change to an alternate message area. Alternate message areas can be handy for storing "file copies" of messages.

There's more information about alternate message areas later in this section.

F10 Move message This is used to move your current message to an alternate message area. The message will be deleted in your current area, and will become the last message in the target area.

Moving a message works just like changing areas, except that you don't change areas, your current message does.

Services

The MAIL service function menu, which is available by pressing F2 from the Utilities menu, provides several special services which will probably be rarely used. The available functions are:

F1 Main menu This is used to exit from the services menu and return to reading messages. Use this if you got the services menu by mistake.

F2 Colors This is used to change the colors used for various things. The color selection screen is fairly self explanatory, and pressing "Alt H" will get you even more help. Your choices for colors will be saved, and used in the future whenever you run MAIL.

F3 Push to DOS This is used to invoke a new generation of DOS. You get a system prompt, and can enter any system command (including running other programs). MAIL will still be sitting in the background, though. Type "EXIT" at the system prompt and you will "pop" back into MAIL, right where you left off.

F4 Manual poll This is used to poll another system on demand. You might want to do this to poll a central system to pick up any mail that it is holding for you.

F5 Alias This is used to select among alternate network addresses. Almost nobody will ever have any use for this.

F6 AutoAlias This is used to turn automatic aliasing on and off. Automatic aliasing means that if you reply to a message which is addressed to one of your alternate addresses, then the reply will be made from that alternate address instead of your primary address. Almost nobody will ever have any use for this.

F7 Show notes This is used to control whether or not you can see extended address fields and other notes which are normally hidden. Messages don't always have hidden notes, and when they do you usually won't want to be bothered with them.

F8 Self mail This is used to control whether or not you can enter a message addressed to your own network address. You normally won't want to do this, but you might if there are several people sharing your network address, or if you are acting as a gateway to a foreign network.

The F3 (Push to DOS) function is *not* the same as a resident program! It is intended to allow you to drop out for a short while only. When you are finished with your mail you should use the F1 (Quit) command to terminate the MAIL program.

The three functions for F6, F7, and F8 all act as "toggle switches", meaning that you press it once to turn it on, and press it again to turn it off. In addition, MAIL remembers how you left these switches, so you don't have to set them again the next time you run MAIL. Most people will not need to bother setting them at all.

F9 Search

The "F9" key is used when you want to search through your messages for something. It brings up a short secondary menu which is used to select what sort of search you want to perform, as follows:

F1 Main menu Go back to reading messages.
F2 Backward Search back through older messages, or
F2 Forward Search ahead through newer messages.
F3 Search name Search for a name from or to.
F4 Search node Search for an address from or to.
F5 Search subj Search for a subject.
F6 Search text Search message text.

The search always starts at your current message. A small green arrow will appear near the bottom right corner of your screen to indicate in which direction

the search will be made. If the arrow points to the left, then the search will proceed through older, "leftward" messages. If the arrow points to the right, then the search will proceed through newer "rightward" messages. Use the "F2" key to turn the arrow around.

When you search for a name, subject, or text you do not have to enter the whole thing, you can enter any part. You do not have to worry about upper case and lower case. "Smith" will match "John Smith", "Smithson John", and "John Whitesmith".

Once you have searched for something, MAIL will remember what it is you searched for last time. For example, if you searched for the name "Smith", and then asked to search for a name again, MAIL will remember that the last name you searched for was "Smith". So if you just press the Enter key it will look for "Smith" again. On many monitors the remembered thing will show up as "shadow text".

10 Select

The "F10" key acts as an index to the messages you have on file. It gives you a listing of messages, one line per message, showing as much as will fit of who it is from, who it is to, and the subject. One of the lines will have a white bar over it (reverse video). The bar starts out on your current message, and can be moved up and down using your up and down cursor movement arrows. You can also move the bar around using the following keys:

PgUp	Moves the bar up several lines.
PgDn	Moves the bar down several lines.
Home	Moves to the first (oldest) message.
End	Moves to the last (newest) message.
.	Moves to the current message.
*****	Moves to the message after the last message read.

You can return to reading messages either by pressing the "F1" (Main menu) key, or the "F2" (Select) key. Pressing "F1" returns you to the message you were reading before you asked to select a message.

Pressing "F2" selects the message under the white bar as your current message, and returns you to it. Pressing the "Enter" key has the same effect as pressing "F2".

You can also use the "F3" key while you are selecting a message. This alternates between showing and not showing the node numbers of who the messages are from and to.

The SEND program

Files are normally sent from one SEAdog to another by attaching them to messages. In this way, the nature and purpose of the files can be explained in the message. In many cases however, it is not necessary to explain the files. In these cases it is possible to send the files without having to invoke the normal user interface.

The SEND program creates null messages (no text) with attached files based on the arguments given in the command line. The format of the command line could be summed up as follows:

```
SEND <files> [CRASH] [HOLD] [NOW] [TO <address>]
```

This is a rather computerish way of stating it, but it's really quite simple. You just follow the word "SEND" with the names of the files to be sent. You can follow the names with the word "CRASH" if the files are to be sent *crash priority*, and you can say who to send them to by adding the word "TO" followed by a name or address at the end of the command.

For example, if you wanted to send a file named READ.ME to node 13/1, you could enter the command:

```
send READ.ME to 13/1
```

This would be done from the DOS prompt ("A>"). If you have a user list file (see the Installation section for details), then you can use a name instead of a net address, like so:

```
send READ.ME to Randy Bush
```

This assumes that Randy Bush is listed in your user directory. If he isn't, then you will be asked what node to send the file to.

You can send more than one file, and you can use path names and wildcards. If you don't use a path name, then your current directory is assumed. SEND will print a warning if the file you are trying to send does not exist.

If you don't specify where to send the file(s), then SEND will ask you. You can then give destination addresses the same way you would when using the user interface, including distribution lists, multiple destinations, and so forth. For example, if you entered:

```
send MYFILE.*
```

then SEND would ask you to enter a net and/or node, exactly the way MAIL does.

There's also a "NOW" keyword which you may find useful for daytime crash mail. "NOW" is similar to "CRASH", and marks the file transfer as crash priority. It also attempts to load and run the MAILER program to send the file immediately. The MAILER program will terminate as soon as it is finished.

For example, if you had a file named "REPORT.WKS" that you had to send to Joe Schmoe right away, then (assuming his MAILER is running) you could type:

```
send REPORT.WKS to Joe Schmoe now
```

You'll need to have a daytime crash mail event set up in order for the "NOW" option to work properly. Please refer to the **Crash mail** section for more details about using crash mail.

There's also a "HOLD" keyword that you can throw in if you want. It means that the file isn't really sent, but is held until the other system calls in to pick it up. This is not the same as a file request, and normal SEAdog users will have no need for it.

If you give the SEND command without any arguments at all, then it will give you a very brief description of how to use it.

The GET program

GET is similar to SEND, except that it enters a file request or an update request. It can be summed up in a similar computerish manner like this:

GET <files> [UPDATE] [CRASH] [NOW] [FROM <addr>]

For example, if you wanted to request a file named NODELIST.BBS from net address 100/76, you would enter the command:

get NODELIST.BBS from 100/76

If you already had a file named NODELIST.BBS, and you wanted to make sure that it was the latest version, you could enter:

get NODELIST.BBS update from 100/76

If you have a user directory, then you could enter that as:

get NODELIST.BBS update from Ben Baker

Assuming, of course, that Ben Baker is listed in your user directory.

If you don't specify FROM who, then GET will ask you. In general, GET is used the same way SEND is.

If you give a path name of a file to request, that doesn't specify where to get it from on the other system. Instead, it says where to *put* it on *your* system. You can only request files which are in a designated pickup area on the other system; you have no control over where they come from.

For example, if you typed:

get A:REPORT.WKS from Joe Bloe

then SEAdog would request the file REPORT.WKS and, if the other system permits it, will put it on drive A: of your system. Similarly, typing:

get \LOTUS\REPORT.WKS from Joe Bloe

causes your SEAdog to request the file and put it in a directory named \LOTUS on your system.

If you do not specify a drive or directory for the file you are requesting, then it goes in your current directory on your current drive. "Current", in this case, means wherever you were when you typed the GET command.

The file and update request functions of MAIL work the same way.

If you enter the GET command without any arguments at all, then it will give you a very brief description of how to use it.

The TELL program

TELL is very similar to SEND, and is used in almost exactly the same way. The difference is that SEND creates a message with a file attached to it, while TELL uses the file to create the text of the message.

In other words, if you type:

```
tell CANNED.TXT to Joe Bloe
```

TELL will create a message to Joe Bloe, where the text of the message is the contents of the file CANNED.TXT. Needless to say, the file must contain normal ASCII text. Use SEND if you want to send a program or a binary file.

If the first character of the file is an asterisk ("*"), then the rest of the first line (up to seventy characters) will be the subject of the message. Otherwise, the subject line is left blank.

If the text in the file has a margin wider than about sixty columns, then it will look odd when read using MAIL. Alternatively, you can have each paragraph of the message stored as one very long line, in which case it will look okay when read using MAIL or any other FidoNet compatible mail system.

Selecting a destination

When you enter a message or ask to send a file, SEAdog needs to know where to send it. When you request a file SEAdog needs to know where to send the request. All of the programs and commands described in this section will, at one point or another, ask you for the destination.

Destinations may be given in any of several ways. Usually you are asked for the *name* of the person to send the message to. If you have a user directory, and if the person's name is listed in the directory, then the message will automatically go to the network address given in the directory.

If you do not have a user directory, or if the other person's name isn't listed, then SEAdog will ask you for their *network address*. This is either a single number, or two numbers separated by a slash. A typical network address might be "375", or "107/7".

If an address is two numbers, then the first number is the *network number*. You can think of network numbers as being similar to telephone area codes. You don't have to give a network number for someone in your own network (though it doesn't hurt). For example, "1/97" means "node 97 in network 1".

Instead of giving someone's name for the destination, you can enter a network address directly. This lets you override the network addresses given in your user directory. If you do, you will be asked again for the name of the person the message is going to.

You can give more than one network address for a message, in which case a copy of the message is sent to each address. You can give as many as you can fit on the line, separated with spaces or commas.

If you enter a slash ("/"), then SEAdog will ask you to select a network. If you enter a number sign ("#"), then SEAdog will show you a list of the nodes in your selected network. These can be combined in one operation. For example, entering "107/#" tells SEAdog to select network 107, and then list all of the nodes in that network.

You can enter the word "ALL", which tells SEAdog to send a copy of your message to every person in your network. This can also be combined with a network number and a slash. For example, entering "107/ALL" tells SEAdog to send a copy of your message to everyone in network 107.

You can enter the name of a file, which should be saved in the SEAdog work area. The file (known as a *mailing list*) should be a normal text file, where each line of the file contains something you might have typed in as a destination. A semicolon anywhere in the file marks the beginning of a comment, and everything from the semicolon to the end of the line is ignored. Here's an example of the kinds of things that can go in a mailing list:

```
1/1                         ;Home office
2/1 3/1 4/1                 ;Branch offices
Ken Kaplan                  ;President
Ben Baker                   ;Treasurer
DIST-B                      ;Another mailing list
```

You can combine any of these in one line, except a person's name. If you give the name of a person, then the name must be the only thing on the line.

For example, you could say to send a message to:

```
375 107/all dist-a
```

This would tell SEAdog to send a copy of your message to node 375 in your own network, everyone in network 107, and everyone listed in your "DIST-A" mailing list.

Carbon copies

Another way of sending a message to more than one destination is to use *carbon copies*.

To send carbon copies, the first line of your message text must be the carbon copy list. This is the word "CC:" followed by a list of where to send the carbons. The "CC:" can be in either upper or lower case. The "CC:" list should be followed by a blank line.

Here's an example of a message with carbon copies:

```
Message #56  0.22
Date: Mon 24 Mar 86 14:06
From: Thom Henderson on 7, SEAboard, Wayne NJ
To:   Gee Wong on 312, Dance Studio, Orange NJ
Subj: Re: Network routing

cc: Rob Elliott, Don Daniels, Karl Schinke

It sounds like a good idea to me.  Let's get
started right away.
```

This message would go to Gee Wong, and copies would be sent to Rob Elliott, Don Daniels, and Karl Schinke. If any of these people are not listed in your user directory, then SEAdog will ask you what address to send the message to.

SEAdog does *not* ask you for the "CC:" list. You enter it as the first line of your message text. It must be the *first* line.

You can specify carbon copies any time you enter a message, including when you reply to another message. Carbon copies are not affected when you edit a message. That is, if you enter a message with carbons going to five people, and then you later edit the original message and make changes to it, the carbons will not be affected. Nor will new carbon copies be sent. A carbon copy list is *only* effective when entering an original message or reply.

Alternate message areas

A message area is nothing more than a directory in which message files are stored. The MAIL program allows you to use alternate message areas. This can be useful for keeping "file copies" of messages that you don't need constantly, but don't want to throw away.

The main message area is the *network mail area*. Only messages saved in the network mail area can be sent to other systems, and messages received from other systems are always placed here. MAIL will always start out looking into the network mail area.

An alternate message area, then, is a directory other than the network mail area in which messages may also be stored.

An alternate message area is created using the MKDIR, or MD, command in DOS. Change directories to the SEAdog work area, and type:

```
MD <name>
```

where <name> is the name of the alternate message area you want to create. Please refer to your DOS manual for more information about creating and using directories.

Multiple message areas are used by the *Change area* and *Move message* functions on the **Utilities** menu.

Both of these functions need you to specify the alternate message area to use. It's done the same way in both cases.

You will get a menu of the available alternate areas, as defined by a file named "AREAS.DOG". This is a normal text file, saved in the same directory as the SEAdog programs and files. Each line of the file defines one alternate message area. Here's an example of what might be in an AREAS.DOG file:

```
D:\MAIL\IFNA        International FidoNet Association
D:\MAIL\REGCON      Regional coordinators conference
D:\MAIL\FSC         FidoNet Standards Committee
D:\MAIL\METRONET    MetroNet sysops conference
D:\MAIL\DOGGIES     FidoClones conference
D:\MAIL\FILE        Old messages on file
```

The first "word" on each line is the name of the directory that the messages are stored in. It is followed by one or more spaces, and then a description of that message area. This description is what is displayed on the message area selection menu, and is also displayed in the upper right corner of your screen whenever you are in an alternate message area.

The description is optional, and MAIL will use the name of the directory if one isn't provided.

When you select an alternate message area, you will be given a menu, where each line of the menu is the description of one message area as given in the AREAS.DOG file. They will be shown in the order they are listed in the file, and only as many of them as will fit on your screen will be shown. They will be numbered sequentially, starting at one.

The first menu line will be numbered zero, and will be your network mail area. This is the only line that will appear if you do not have an AREAS.DOG file.

One message area will be highlighted by a reverse-video (black on white) bar. This is your current message area.

You can select an alternate message area in any of three ways:

1) Use your cursor movement arrows to move the bar up and down. Place the bar on the message area you want, and press "Enter".

   The up arrow moves the bar up one line, while the down arrow moves it down one line. The "Home" key moves it to the first line (your network mail area), and the "End" key moves it to the last area.

2) Type the number of a message area, and press "Enter". The white bar will move directly to that message area to show what you selected.

3) Type the name of any directory anywhere on your system, and press "Enter". This allows you to use any directory at all, including ones not listed in your AREAS.DOG file. Using this method, you don't even have to have an AREAS.DOG file (though it does make life simpler).

If you decide that you do not want to change areas or move a message after all, just press the "F1" key, and you will return to reading the messages in your current area.

MAIL will allow you to enter a message in an alternate message area, but it will warn you about it. Mail entered in alternate message areas is *not* sent to other systems, so if you enter a message in an alternate area you will not be asked for the destination address. Also, default message bits do not apply in an alternate message area.

THE ROBOT MAILER

Robot is designed to automate file distribution within the SEAdog electronic mail system.

Robot works by creating null messages (no text) with attached files or file requests. It has its own control file, named ROBOT.DOG, which tells it when to create its messages. It should be run daily as an external event shortly before network mail is processed.

ROBOT.DOG is a text file which can be created using any standard editor. Each line of the file specifies one "action set". That is, a file or a group of files and a node or a group of nodes to send it to (or request it from), along with time specifiers that say when the file is to be sent (or requested).

This is all much easier than it sounds. Hopefully, it will be intuitively obvious what is going on.

A sample entry might be:

```
Sat NODELIST.?Q? 107/1 107/2 107/3 107/4
```

This example starts with a time specifier, "Sat", followed by a filename template, followed by a list of nodes to send to. This would cause four outgoing messages to be created when Robot is run on a Saturday.

Time can be specified as a three-letter day of the week, as the keyword "ALL" (for every day), as an integer (for a day of the month), or as two integers separated by a slash (for a day of the year). For example:

All	Means do it every day
Tue	Means do it every Tuesday
15	Means do it on the 15th of every month
12/13	Means do it every December thirteenth
2/29	Means do it every leap day

There are several keywords which modify how Robot creates messages for the filenames which follow them. They are:

REQUEST	This tells Robot to generate a file request.
UPDATE	This tells Robot to generate a file update request.
EXIST	This tells Robot that it should only generate a message if a template matches one or more existing files.
CHANGED	This tells Robot that it should only generate a message if a file exists and has been modified, as shown by its archive bit.
CHANGED-<n>	This tells Robot that it should only generate a message if a file exists and has been modified within the last <n> days.
HOLD	This tells Robot that the message it generates should be held for pickup by the other system. This will only work if you have a low security mail system. Please refer to the **Low security mail** section for more details about mail pickup and polling.
CRASH	This tells Robot that the message it generates should be marked crash priority. Please refer to the **Crash mail** section for more details about crash priority mail.
TEXT <file>	The TEXT keyword is followed by the name of a file, which should contain standard ASCII text. The contents of this file is inserted in each generated message as the text of the message. This remains in effect for all following lines until another TEXT keyword, or a NOTEXT keyword, is reached. If the

named file does not exist, then no text is inserted in the generated messages.

NOTEXT — This tells Robot to stop putting text in its messages.

KILL — This tells Robot to delete any messages which were created by Robot. All Robot generated messages are deleted, even ones from other systems.

File names should be given as path names (based on the SEAdog base directory), and may include wildcards. File names which follow a CHANGE or CHANGE-<n> keyword may not include wildcards.

Node numbers can be given as either a node, or a net/node. In the former case your own net is assumed.

As usual, a semicolon starts a comment, and causes the rest of the line to be ignored.

Special notes

At this point you cannot send or request files named REQUEST, UPDATE, and so forth; but you can get around that by appending a period. In other words, "UPDATE." will send a file, "UPDATE" will not.

The name of the file to send cannot begin with a digit. If it does, then Robot will try to interpret it as a node number. This can be gotten around in either of two ways:

```
Mon ?1070007.USR   107/1          ;Use a wildcard, or
Mon .\01070007.USR 107/1          ;Use a path to avoid it.
```

When you use the CHANGED keyword, Robot uses the MS-DOS archive bit to determine if a file has changed. Robot will reset this bit. You should be aware of this if you use the /m option of the BACKUP command, or if you take daily backups of your system.

Robot uses the MS-DOS date stamp to determine the number of days since last change for the CHANGED-<n> keyword, except that CHANGED-0 is equivalent to CHANGED. File names to the left of a CHANGED keyword may not include wildcards.

Here are some examples of entries using CHANGED keywords:

```
Mon changed   XYZ.ARC 107/1          ;Send XYZ.ARC only if changed
Mon changed-7 XYZ.ARC 107/1          ;Or only if changed this week.
Mon ABC.ARC changed XYZ.ARC 107/1    ;Always sends ABC.ARC
Mon changed   XYZ.*   107/1          ;This won't work
```

Robot in the demand mode

All of the above describes how to use Robot in its "native mode". This is intended to work as a once a day external event, which is run shortly before mail hour, and can handle all regularly scheduled file mailings.

But in some cases you may have a standard set of file mailings that do not happen on a regular schedule. An example of this might be routing files which are only sent out when a routing change occurs.

Robot can also handle this situation with aplomb. It is done by using "prefix labels" instead of time specifiers. Here are some examples of prefix labels:

```
Route: ROUTENJ.ARC 107/16          ;New Jersey hub
Route: ROUTENY.ARC 107/3           ;New York hub
Route: ROUTEDE.ARC 107/50          ;Delaware hub
```

You will note that the above examples don't give a day of the week. Instead, each line begins with a label that ends with a colon.

When Robot is run in its native mode it takes no arguments, and ignores all lines with prefix labels. When Robot is run in demand mode it is given at least one argument, and performs only those mailings where the prefix label matches the first argument (case is not important). For example, when you wanted to

perform the mailings in the above example, you would type the command:

```
robot route
```

You can also give extra arguments in demand mode, which are used to perform substitutions in the same manner as DOS batch files. For example, consider the following:

```
node: nodelist.%1 101/0
node: nodelist.%1 102/0
```

In this case, the intent is to send out a node list on demand, where the full name of the nodelist changes from time to time. You would send the nodelist for day 263 as follows:

```
robot node 263
```

This would cause Robot to create messages to send NODELIST.263 to nodes 101/0 and 102/0. You can specify up to nine arguments, designated as %1 through %9.

A sample Robot control file

Here is an example of a control file for Robot:

```
Sat kill                              ;Delete old messages
Sat           MASTER.WKS  13/10      ;Send in master spreadsheet
Sat exist     WEEKLY.REP  13/10      ;Send this only if there
Sun request   WEEKLY.SUM  13/10      ;Get summary report

Sat changed-7 BUDGET.DAT  13/11      ;Send this only if changed
Sun update    BUDGET.MAS  13/11      ;Get this only if changed

Wed text      BRAG.TXT              ;Text for Wednesday
Wed exist     PROFIT.REP  13/1      ;Send file and message text
Wed exist     PROFIT.REP  13/2      ;Send file and message text
Wed exist     PROFIT.REP  13/3      ;Send file and message text
Wed text      EXCUSE.TXT            ;New text for Wednesday
Wed exist     LOSS.REP    13/1      ;Send file and message text
Wed exist     LOSS.REP    13/2      ;Send file and message text
Wed exist     LOSS.REP    13/3      ;Send file and message text
Wed notext                          ;Stop using message text

Thu hold exist BALANCE.WKS 13/4     ;Hold for pickup
```

THE TWIX MAIL PRINTER

The TWIX program is used to print out your mail on your printer. Obviously, you need a printer to do this. This can also be done from the MAIL user interface program. TWIX is provided so that you can use external events to have your mail printed out automatically.

How TWIX works can be modified by giving it any of the following arguments:

KILL This tells TWIX to kill (ie. delete) each message it prints. This can be handy if you don't want to keep old messages "on file", and want them cleaned up automatically.

TRANSIT This tells TWIX to print out all messages on your machine, even ones which are not addresses to you (such as messages being routed through your node). Messages in transit are never killed by TWIX. Normally, you should never need to do this.

UNREAD This tells TWIX to treat the messages it prints as if they have not yet been read.

ALL This tells TWIX to print out all of your mail, even old messages which you have already seen.

HELP This tells TWIX that you need help. TWIX responds by giving you a brief description of how to use it.

TWIX is mainly intended to print out your mail for you in the morning. To do this, it needs to be configured as an *external event*. Refer to the **Installation** section for details on external events.

THE USER LIST

In a large network (or even in a small one) it can be difficult to remember everyone's network address. The usual solution is to use some sort of directory. SEAdog can also use a directory, but unlike other systems, SEAdog can look names up in it for you.

A SEAdog directory is called a *user list*, and is nothing more than a sorted list of names with their associated network addresses, all stored in a normal text file. SEAdog can use two user lists at the same time. They are called:

    USERLIST.DOG
    FIDOUSER.LST

Whenever SEAdog wants to know what network address is associated with any given name, it will search for it in the first user list (if any). If it doesn't find the name listed there, then it will search the second user list (if any). If SEAdog still can't find the name listed anywhere, then it will ask you for the network address.

The intent of this is that the second list can be used as a standard network "phone book" distributed with the node list, while the first list can be used as a "nickname file" or as a list of exceptions. But you don't need to use two lists if you don't want to. Indeed, you don't need to use any user list at all if you don't want to.

SEAdog can be pretty fussy about the format of a user list. You can use any normal text editor (such as EDLIN) to create and maintain one, but you're probably better off using ULMAINT. Later on we'll describe how a user list is put together, just in case you want to create a user list some other way.

Using ULMAINT to maintain the user list

The ULMAINT utility is especially designed to help you maintain a user list in the proper format. You can use it either to create a list from scratch, or to repair and reformat a list created some other way.

Start ULMAINT by typing the word "ULMAINT" (without the quotes) at the DOS prompt. The first thing it will do is to make a copy of your existing user list (if any), ensuring that all of the lines are the same length.

It will then ask you for a name. Enter the name of a user, as you would if you were addressing a message to them.

If the user is already listed, ULMAINT will ask you if you want to change their address. Press the "Y" key to enter a new address for that user, or the "N" key to leave their address unchanged.

If you press "N" (to leave the user unchanged), ULMAINT will ask you if you want to delete the user. Press "Y" to remove that user from your user list, or "N" to keep him. If you press "N", then ULMAINT will ask you for the name of another user.

If you are adding a new user or changing an existing user, ULMAINT will ask you for their network address. You can enter anything that would be legal when selecting a node to send mail to, including any form of extended addressing. If you enter a blank line, then ULMAINT will ask you for the name of another user.

Exit ULMAINT by entering a blank line instead of a name. ULMAINT will then consolidate all of your changes and generated a sorted user directory. A copy of DOS SORT must reside on your command path somewhere, or ULMAINT will complain.

User list format

This section is included for those who may wish to create a user list by some means other than using ULMAINT. This could mean anything from doing it by hand with an editor, to having some data base manager generate a list, to almost anything else you might want. Most people won't need to worry about this.

A user list is a standard text file which can be created and maintained using any normal text editor (such as EDLIN), but you must be careful. Each line defines the entry for one person. The line begins with the person's name (last name, comma, space, first name), and the line ends with a network address for that person.

The file must be sorted by ascending order of names, and every line of the file must be *exactly the same length*. It doesn't matter what the length is, as long as it is the same for every line.

Names must be given with the first character of each word in uppercase, and the rest in lowercase. For example, "John McGill" would be stored as "Mcgill, John", while "Randy van de Loo" would be stored as "Loo, Randy Van De".

Here's an example of what might be in a typical user list:

```
Foray, Andy                 107/7
Hartman, Bob                132/101
Henderson, Thom             107/8
Horowitz, Dave              107/2
Wong, Gee                   107/312
```

This is fairly straightforward, and should not be too difficult to manage. Just remember that the file *must* be sorted alphabetically, and each line *must* be the same length (including trailing spaces, if any) for this to work properly.

Some editors and programs leave a "control Z" at the end of a file after you've edited it. You shouldn't have that, or the user list won't work. Luckily, there's a simple way to fix it. The DOS SORT command will remove the control Z and any trailing garbage from your user list, and make sure it's properly in order to boot. So when in doubt, filter the user list through the SORT command.

NODE LISTS

If your SEAdog is going to send mail to any other systems, then it needs to know certain things about them, such as their phone number, what baud rates they can support, and so on. This information is supplied by the node list.

The node list is initially fed to SEAdog as a file named NODELIST.BBS. SEAdog will scan this list once to create its own format node list, and then again whenever the node list changes.

As a general rule, a network will have a *network coordinator* who maintains the node list for the entire network. Most users don't need to worry about the format and content of the node list; they just install the node lists as they are distributed. (This can even be automated by using Robot to get the updates.) Only the network coordinator really needs to read this section.

NODELIST.BBS is a standard text file which can be created and maintained by any normal text editor (such as EDLIN). Each line of the file defines one node, and contains the following (in order) separated by spaces:

1) The node number of this node.

2) The estimated cost to send one message to this node, in pennies.

3) The maximum baud rate supported by this node.

4) The name of this node (up to 14 characters).

5) The phone number for this node (up to 40 characters).

6) The city and state where this node is located (up to 40 characters).

As usual, a semicolon marks the beginning of a comment, and everything from the semicolon to the end of the line is ignored.

Here is an example of a few node list entries:

```
;   Sample NODELIST.BBS for a basic SEAdog net
;
1 22 1200 SEAdog_Leader  1-201-478-8201 Clifton_NJ
2 22 1200 SEAboard       1-201-694-3348 Wayne_NJ
3 22  300 SEAnet_Relay   1-201-478-1633 Clifton_NJ
```

As you can see, this isn't really very hard. There are only a few things to keep in mind:

1) Since spaces are delimiters, you cannot put any spaces in any of the names, phone numbers, etc. Use underlines instead ("_"). They will be translated into spaces when the names are displayed.

2) You should not use commas anywhere either. They are mainly needed in phone numbers to tell a modem to delay a bit. In this case, you can use periods instead. SEAdog will translate periods into commas when it dials the phone.

3) Your phone numbers should be stored in a consistent form, or else the DIAL configuration statement won't work properly.

4) You can't leave anything out.

A node list like the one above defines a *basic point-to-point net*. This means that if you enter a message to each of five other nodes, then your system will make five phone calls. If your network contains five nodes, and each node has a message for every other node, then a total of twenty calls are going to be made. This isn't too bad, but if you have fifty nodes all sending mail to each other, then the total number of calls goes to 2450. And for a hundred nodes, it shoots up to almost ten thousand phone calls! That's an awful lot of phone calls.

Now in any given situation, you probably won't have every node calling every other node every night, so the numbers won't get as bad as this, but it still serves to illustrate the problem.

The basic point-to-point net is suitable for up to five or ten nodes, and can be used on larger nets

where the overall mail traffic is light. When you get more nodes, or when clusters of nodes are in widely scattered locations, then you'll want some sort of network routing.

Basic net with hubs

There are several things you can do. The easiest is to establish a *basic net with hubs*. This is done by adding routing prefixes to the node list. Here's an example of a node list for a basic net with hubs:

```
;    Sample NODELIST.BBS for a basic SEAdog net with hubs
;
HUB 1 22 1200 SEAdog_Leader  1-201-478-8201 Clifton_NJ
    2 22 1200 SEAboard       1-201-694-3348 Wayne_NJ
    3 22  300 SEAnet_Relay   1-201-478-1633 Clifton_NJ
HUB 4 22 1200 Operations     1-609-698-3377 Waretown_NJ
    5 22  300 Marketing      1-609-693-1055 Manahawkin_NJ
    6 22 2400 Customer_Svc   1-609-693-8055 Ship_Bottom_NJ
```

This is just like the earlier example, except that we've put the word "HUB" in front of two of the entries. This tells SEAdog that those two nodes are *routing hubs*. When a node is marked as a hub, he then handles all mail traffic for all of the nodes which follow him in the node list, down to the next HUB prefix.

In other words, in the above example the first three nodes send all of their mail to node 1, no matter who it is really addressed to. Node 1 then passes it on to its destination. Node 1 will also bundle up all the mail from any of the first three nodes to any of the last three in one packet, and give it to node 4. Node 4 will then pass it along. (This works the other way as well, of course.)

This sounds complicated, but it really isn't; and the advantages are enormous. As an example, consider node 3. Without hubs, it might have to make as many as five phone calls a night (one to each other node). But now it only makes one call a night, to its hub. In the basic net as many as thirty phone calls could be required on a given night. But with hubs only twelve phone calls are needed to pass mail from every node to every other node. With larger nets the improvement is even more dramatic.

The only problem with all this is that a message only makes one "hop" in any given mail event, so you have to add more mail events. In a basic net with hubs you'll need at least three mail events.

Consider a message from node 2 to node 5. First node 2 sends it to its hub, node 1. Then node 1 sends it to the destination hub, node 4. Then node 4 passes it along to its final destination, node 5. This is a total of three hops, so at least three mail events are needed. (In practice, only the hubs need more than one mail event, but it's best to be consistent.)

Here's an example of how to set up the mail events for a hubbed net:

```
Event H All 2:00 3:00        ;Local nodes upload to hubs
Event T All 3:00 4:00        ;Hubs call other hubs
Event L All 4:00 5:00        ;Hubs download to local nodes
```

Note that we have chosen some rather odd tags for these events. Why not just tag them *A*, *B*, and *C*? Different routing tags mean different things, and each one tells SEAdog what to do with your mail in any given mail event for the greatest savings. The exact meaning of any given tag is defined in the **Routing Tags** section later on. For now, just use these three tags, in this order, for a basic net with hubs.

Multiple nets

There is quite a bit you can do in addition to using hubs. If you have a *very* large net (Several hundred nodes), then you might want to split it up into several smaller nets. Each node gets a *net number* in addition to its node number. All of the nodes in all of the nets can send mail back and forth, since they all use the same node list. You can think of the net numbers as being similar to telephone area codes.

In addition, node numbers can be "re-used". You can have a "node 1" in every net, as long as they have different net numbers. Multiple nets ("multinets") are usually used to split up the administration of a large mail system. Each of the local nets can have its own coordinator who assigns node numbers in his own net and maintains his piece of the overall node list. An overall *system coordinator* then combines all

of the individual pieces to create a master node list, which is then distributed.

Here's an example of a node list for a multinet system:

```
;    Sample NODELIST.BBS for multinet SEAdog
;
;    Network 1 -- Northern New Jersey
;
NET 1 SEAnet_East    Clifton_NJ
HUB 1 22 1200 SEAboard        1-201-694-3348 Wayne_NJ
    2 22  300 SEAnet_Relay    1-201-478-1633 Clifton_NJ
    3 22 1200 SEAdog_Leader   1-201-472-8065 Clifton_NJ
;
;    Network 2 -- Southern New Jersey
;
NET 2 SEAnet_South   Waretown_NJ
HUB 1 22  300 Marketing       1-609-693-1055 Manahawkin_NJ
    2 22 2400 Customer_Svc    1-609-693-8055 Ship_Bottom_NJ
    3 22 1200 Operations      1-609-429-6630 Waretown_NJ
```

This is very similar to a basic net with hubs, except that we've added the *NET* statement. A NET statement tells SEAdog that all of the nodes which follow it (down to the next NET statement) are part of a given net. A NET statement contains the following (in order) separated by spaces:

1) The keyword "NET".

2) The net number for this net.

3) The name of this net (up to 14 characters).

4) The city and state where this net is located (up to 40 characters).

The above example gives a node list for a *multiple net with hubs*. If you have a large volume of traffic going between the different nets, then you may want to split up the duties of the hubs.

Multiple nets with gateways

Each hub handles all the mail *to* or *from* all of his nodes. If you have enough mail moving around, a hub can get "mobbed", either not having enough time to

send all his mail, or being too busy for other hubs to reach, or both. In this case, you would want to split up his duties and establish *inbound and outbound gateways* for his net.

An *inbound gateway* acts like a hub for all mail addressed to his net. Any node having mail for anyone in the net will give it to the inbound gateway.

An *outbound gateway* handles all outbound mail. Any node in a net with an outbound gateway will give his mail to the outbound gateway.

By setting up inbound and outbound gateways, overall performance is greatly enhanced. The inbound gateway doesn't place any calls himself, so he is always free to take calls. Nobody calls the outbound gateway, so he is free to make calls.

Here's an example of a node list for a multinet system with gateways:

```
;    Sample NODELIST.BBS for multinet SEAdog with gates
;
;    Network 1 -- Northern New Jersey
;
NET   1 SEAnet_East     Clifton_NJ
      1 22 1200 SEAboard         1-201-694-3348 Wayne_NJ
IGATE 2 22  300 SEAnet_Relay     1-201-478-1633 Clifton_NJ
OGATE 3 22 1200 SEAdog_Leader    1-201-472-8065 Clifton_NJ
;
;    Network 2 -- Southern New Jersey
;
NET   2 SEAnet_South    Waretown_NJ
OGATE 1 22  300 Marketing        1-609-693-1055 Manahawkin_NJ
      2 22 2400 Customer_Svc     1-609-693-8055 Ship_Bottom_NJ
IGATE 3 22 1200 Operations       1-609-429-6630 Waretown_NJ
```

In this example, node 1/3 is the outbound gateway for net 1, and node 2/3 is the inbound gateway for net 2. If node 1/1 wanted to send a message to node 2/1, he would send it to node 1/3, who would pass it to node 2/3, who would give it to node 2/1.

As with a basic net with hubs, three mail events will be needed to get the mail through overnight. Here is an example of the mail events to use with a multiple net with gates:

```
Event G All 4:00 5:00        ;Locals upload to outbound gate
Event T all 4:00 5:00        ;Outbound gates call Inbound gates
Event L all 5:00 6:00        ;Inbound gates download to locals
```

Three tiered mail systems

If you have a *very* large system (hundreds of nodes and heavy mail traffic), then you can combine hubs and gateways to create a *three tiered mail system.* A three tiered system can handle an incredible volume of mail traffic and still provide overnight mail delivery.

Here's an example of a node list for one net in a three tiered mail system using hubs and gateways:

```
NET          107 NYC_Metro  New_York_NY
HUB          134 0 2400 AMuse                  1-212-269-4879 New_York_NY
              18 0 2400 NYU_Med_Center         1-212-889-7022 New_York_NY
              31 0 2400 Rainbow_Corner         1-914-425-2613 Spring_Valley_NY
              35 0 2400 PHALSE_Fido            1-914-472-6522 Scarsdale_NY
              37 0 1200 HumanTech_CUSSNET      1-212-532-2278 New_York_NY
             101 0 1200 MainFrame_Connect      1-212-627-2673 New_York_NY
             102 0 2400 BillBoard              1-212-333-3285 New_York_NY
             103 0 2400 APL_PI                 1-212-753-0888 New_York_NY
             104 0 1200 Eastern_Queens         1-718-343-2185 Queens_NY
             105 0 2400 NY_Transfer            1-718-442-1056 Staten_Island_NY
             110 0 1200 Emergency_Medic        1-718-238-8120 New_York_NY
             132 0 1200 NoVaSys                1-212-304-8553 New_York_NY
IGATE HUB    210 0 2400 D2-FIDO                1-516-682-8525 Woodbury_NY
             207 0 1200 Grummans-Fido          1-516-575-5838 Bethpage_NY
             211 0 2400 Daniels-FIDO           1-516-367-9626 Melville_NY
             222 0 1200 Gateway_NRA            1-718-338-3501 Brooklyn_NY
             234 0 1200 Future_Watch_Webb      1-516-265-6333 Smithtown_NY
             240 0 2400 AU_Soc_Svc_Ctr         1-516-228-7498 Garden_City_NY
             269 0 1200 Utopian_Quest_LI       1-516-842-1712 Bellmore_NY
             293 0 1200 BaphoNet               1-718-499-9277 Brooklyn_NY
HUB          317 0 1200 Dumps_R_Us             1-201-885-7404 Piscataway_NJ
               7 0 2400 SEAboard               1-201-472-8065 Clifton_NJ
OGATE         16 0 1200 Wizards_Tower          1-201-288-9076 Teterboro_NJ
              27 0 2400 The_Wizard's_BBS       1-201-379-2185 Springfield_NJ
              29 0 1200 Spider's_Web           1-201-782-7640 Flemington_NJ
             310 0 2400 Cloudbase              1-201-292-1365 Morris_Plains_NJ
             311 0 1200 The_Xtra_BBS           1-201-284-3916 Clifton_NJ
             312 0 2400 Dance_Studio           1-201-249-1898 E_Brunswick_NJ
             313 0 1200 Cork_Board             1-201-943-2226 Ridgefield_NJ
             316 0 1200 Metatek_Fido           1-201-286-2567 Toms_River_NJ
             319 0 2400 The_Micro_Room         1-201-245-6614 Roselle_NJ
             320 0 1200 RisQue_II              1-201-923-9792 Newark_NJ
             321 0 1200 The_Console_Room       1-201-827-7815 Ogdensburg_NJ
             322 0 2400 REACT_System_1         1-201-628-7287 Mountain_View_NJ
             323 0 1200 Arco_Info_Board        1-201-340-2100 Clifton_NJ
             324 0 2400 TMMNET                 1-201-326-9870 Parsippany_NJ
             399 0 1200 Crestwood              1-201-885-7404 East_Amwell_NJ
              17 0 2400 DEC-House              1-609-866-9481 Cherry_Hill_NJ
             414 0 1200 Pinelands_BBS          1-609-354-9259 Cherry_Hill_NJ
              25 0 2400 Softshop               1-215-663-1487 Ocean_City_NJ
HUB          801 0 1200 Spindle_City_FIDO      1-518-235-1179 Cohoes_NY
             701 0 1200 The_IBM_Temple         1-716-838-2664 Buffalo_NY
             703 0 2400 I-Tech1                1-716-874-6509 Buffalo_NY
             704 0 2400 Modem_Madness_BBS      1-716-889-2016 Chili_NY
             705 0 1200 Southern_Tier_FIDO     1-607-772-8024 Binghamton_NY
             706 0 1200 ACE_Base               1-716-626-4925 Williamsville_NY
             708 0 1200 Infancy_Research       1-716-244-7418 Rochester_NY
             723 0 1200 Hitch_Hikers_Guide     1-315-589-7361 Williamson_NY
             810 0 2400 Fly_by_Night           1-518-477-8260 East_Greenbush_NY
```

Note that node 210 has both IGATE and HUB in front of his listing. This means that he is the hub for the nodes which follow him (Long Island), and also the inbound gateway for his net. You can combine the HUB, IGATE, and OGATE prefixes in any manner you wish.

A three tiered mail system requires five mail events per night. Here is an example of the mail events needed to support such a system:

```
Event H all 2:00 3:00        ;Local nodes upload to hubs
Event G all 3:00 4:00        ;Hubs upload to outbound gateway
Event T all 4:00 5:00        ;Outbound gates call inbound gates
Event I all 5:00 6:00        ;Inbound gates download to hubs
Event L all 6:00 7:00        ;Hubs download to local nodes
```

Important note:

No matter what the network structure is, some messages are *never* routed. Any message which has a file attached to it is always sent directly from origin to destination. File requests and update requests are also not routed.

There are a few good reasons for this. The main one is that, while passing around messages is usually pretty fast, it can take a long time to transfer a large file. This could cause a serious degradation of network performance if file attaches were to be routed.

Also, messages which are marked *crash priority* are not routed. The whole point of crash priority is to get it there fast, regardless of cost. For this reason, crash priority mail is always sent direct, point to point.

ROUTING TAGS

The previous section described at length how a sophisticated store-and-forward mail system can be configured by making notes in the node list and by setting up certain mail events.

We didn't explain there what the different route tags are for. We'll do that now.

The single character tag associated with every mail event is known as a *routing tag*. A routing tag specifies how messages are to be routed and sent during any given mail event. The node list defines the structure of your mail system, while the routing tags define how to implement that structure.

Different routing tags mean different things, and you can modify what a tag means any way you like (see the **Advanced Routing** section for more details). Here are the routing tags that SEAdog already knows about:

Tag	Intended purpose
H	Upload to hubs.
G	Upload to outbound gateways.
T	Primary mail event.
I	Download from inbound gateways to hubs.
L	Download from hubs.
S	Daytime crash mail.
A,V,W	Public amateur net compatibility.

The H tag (hubs)

During any mail event with an **H** tag, SEAdog will only call your hub (if you have one).

If your node is a hub or an outbound gateway, then no mail will be sent during this event.

If your node is an inbound gateway, then all mail addressed to nodes in other nets will be routed to your hub. All mail addressed to nodes within your own net will not be sent during this event.

Otherwise, all of your outgoing mail is routed to your hub.

The G tag (gates)

During any mail event with a G tag, SEAdog will only call your outbound gateway (if you have one).

If your node is an outbound gateway, then no mail will be sent during this event.

If your node is a hub, then any mail addressed to any node you are a hub for is not sent during this event. All other mail is routed to your outbound gateway.

If your node is an inbound gateway, then all mail addressed to nodes in other nets will be routed to your outbound gateway. All mail addressed to nodes within your own net will not be sent during this event.

Otherwise, all of your outgoing mail is routed to your outbound gateway.

The T tag (transit)

During any mail event with a T tag, SEAdog will call any node which it is not an inbound gateway or a hub for.

Every message which is addressed to a node that has an inbound gateway is routed to the inbound gateway.

Otherwise, if the destination node for the message has a hub, then it is routed to the hub.

Otherwise, it is not routed.

The I tag (inbound)

During any mail event with an I tag, SEAdog will call any hub.

Every message which is addressed to a node that has a hub is routed to the hub. Otherwise, it is not sent.

The L tag (local)

During any mail event with an **L** tag, SEAdog will call any node, and every outgoing message is sent directly to its destination.

The A tag (All)

During any mail event with an **A** tag, SEAdog will call any node.

Every message which is addressed to a node that has an inbound gateway is routed to the inbound gateway.

Otherwise, if the destination node for the message has a hub, then it is routed to the hub.

Otherwise, it is not routed.

The S tag (Send only)

The **S** tag functions identically to the **A** tag, except that mail is sent in *send only mode*.

SEAdog normally pauses between placing calls for anywhere from one minute to three minutes to allow other nodes to call in. When SEAdog is operating in send only mode this pause drops to a matter of a few seconds. Outgoing mail goes much faster, but it will be almost impossible to receive any calls.

You should not normally use this during normal mail hours unless you are the outbound gateway for your net, since it will be almost impossible for your system to receive any mail.

The **S** tag is mainly intended for use with *crash mail*. Refer to the **Crash Mail** section for more information.

The V tag

During any mail event with a **V** tag, SEAdog will only call those hubs which you are the inbound gateway for.

Every message which is addressed to a node that has a hub will be sent to the hub. No other mail is sent.

If you are not an inbound gateway, then no mail will be sent during this event.

This routing tag is intended primarily as a version of the I tag for use on the public amateur net, which does not support redundant backup.

The W tag

During any mail event with a **W** tag, SEAdog will only call those nodes which you are the hub or the inbound gateway for. Mail is not routed during this event, but is sent directly to its destination.

If your node is not an inbound gateway or a hub, then no mail will be sent during this event.

This routing tag is intended primarily as a version of the L tag for use on the public amateur net, which does not support redundant backup.

Other tags

During a mail event with any other routing tag, SEAdog will not call any other node unless told to do so by its routing file. See the **Advanced Routing** section for more information about routing files.

Meanwhile, the default host and hub routing can be useful, but is tedious to express in a routing file. Therefore, SEAdog performs certain default routing in any mail event as follows:

Every message which is addressed to a node that has an inbound gateway is routed to the inbound gateway.

Otherwise, if the destination node for the message has a hub, then it is routed to the hub.

Otherwise, it is not routed.

If you do not wish to make use of this default routing, then your route file *must* contain the statement:

```
No-route All
```

Suggested use of route tags

We recommend that you use the **H, G, T, I, and L** tags to establish five mail events, as follows:

```
Event H all 2:00 3:00                ;Locals to hubs
Event G all 3:00 4:00                ;Hubs to outbound gates
Event T all 4:00 5:00                ;Outbound gates to Inbound gates
Event I all 5:00 6:00                ;Inbound gates to hubs
Event L all 6:00 7:00                ;Hubs to locals
```

If you intend to use daytime crash priority mail, then you should add:

```
Event S all 9:00 20:00 Crash Dynamic ;Daytime crash mail
```

See the **Crash Mail** section for more information about crash mail.

The times given here are expressed in local time, as given by the system clock, and are somewhat conventional for systems on Eastern Standard Time. You can use whatever times you like, but be sure that all nodes in your system agree as to when the different mail events occur.

For example, 4:00 Eastern Standard Time is 1:00 Pacific Standard Time, so a node in Los Angeles would use:

```
Event H all 23:00 24:00              ;Locals to hubs
Event G all 0:00 1:00                ;Hubs to outbound gates
Event T all 1:00 2:00                ;Outbound gates to Inbound gates
Event I all 2:00 3:00                ;Inbound gates to hubs
Event L all 3:00 4:00                ;Hubs to locals
Event S all 6:00 17:00 Crash Dynamic ;Daytime crash mail
```

If you are setting up a basic net (without hubs or gates) then setting up all five of these mail events will surely seem like overkill. But having them in place and recognized by every node will enable you to change your network structure however you like, merely by changing your node list. Each of the nodes in your net will already know exactly what to do.

Redundant backup

In addition to providing overnight mail service at low cost, the above set of five mail events also provides a great deal of redundant backup to ensure that the mail is delivered promptly.

As an example, let's consider one node in a three tiered mail system who has several messages going out (with no attached files). In the usual case he places one phone call to his hub and hands over all of his mail. He makes no more phone calls that night.

However, suppose his hub is down (perhaps the phone is off the hook). Since he can't reach his hub, then he will end the first event with his mail still unsent. During the second event he will try to call his outbound gateway, and again pass over his mail.

But suppose his outbound gateway is also down (perhaps a fuse blew out somewhere). Since he can't reach his outbound gateway, he will end the second event with unsent mail. During the third event he will try to call each of the inbound gateways for any net he wants to send mail to. In each case he gives the inbound gateway all the mail he has for that net.

But suppose he couldn't reach one of the inbound gateways (perhaps the cleaning lady turned off the machine). Since he can't reach the inbound gateway, he will end the third event with some of his mail still unsent. During the fourth event he will try to contact the hubs of any nodes he still has mail for.

But suppose he couldn't reach one of the hubs (perhaps vandals broke in and smashed the machine). Since he can't reach one of the hubs, he will end the fourth event with a little mail *still* unsent. During the fifth event he will bypass everyone and go straight to the destination for each of his remaining messages.

But suppose he couldn't reach one of the nodes (perhaps an earthquake swallowed the building). Since he can't get through, then he winds up with a few messages still unsent, but it doesn't matter because the destination wouldn't have gotten any mail anyway.

A word about time

As we mentioned earlier, you should try to have all of the nodes in your net start their mail events at the same time. For example, if all of your east coast nodes run event T at 4:00, then all of your west coast nodes should run event T at 1:00. The times given in our earlier example were chosen mainly because they allow prodigious amounts of mail to be moved overnight, while remaining in the "late night" billing category for phone calls in all United States time zones.

If you are running an international mail system, then you may want to adjust these times somewhat, or possibly use different times for mail events in other countries. SEAdog can receive mail at any time, so you can be pretty flexible.

SEAdog is pretty forgiving about time, so if all of the nodes in your mail system have their clocks set slightly differently, it won't matter too much. A few minutes either way should make no difference. If one node is off by an hour or more, the mail will still get through, but it may cost a little more or take a little longer.

CRASH MAIL

A few things have been said about *crash priority mail*. We should probably explain what this is all about.

The intent of crash priority is to have some way of marking urgent messages so that they will be sent as fast as possible, regardless of cost. Once a message has been marked crash priority, it is handled differently in a couple of ways.

First and foremost is that crash mail is never routed. It is always sent directly from source to destination. This avoids any possible chance of it being delayed in network "hangups". Also, some nets are not designed to deliver all mail overnight (though most are).

Crash mail can also become "instant mail" if crash mail events are used. Here's an example of a CONFIG.DOG entry for a crash mail event:

```
Event S all 8:00 17:00 Crash          ;Daytime crash mail
```

This is very similar to any other mail event, except that the keyword *crash* is added to the end. This marks the event as a *crash mail event*. Only crash mail is sent during a crash mail event. All other mail will wait for the late night mail event, when phone rates are lower.

In addition, we have used the special routing tag *S* on this event. This tells SEAdog that it can call any other node, and that it should not pause between attempts.

You can also make the crash mail event *dynamic*. This means that the mail event ends as soon as all of the mail is sent. Here's an example of a CONFIG.DOG entry for a dynamic crash mail event:

```
Event S all 8:00 17:00 Crash Dynamic    ;Daytime crash mail
```

Here's an example of how you might use daytime crash mail in the real world:

Your phone rings. It's Joe Schmoe in LA, and he tells you that he needs to see the XYZ spreadsheet right away. You tell him, "Okay, start up SEAdog!"

He hangs up and types:

```
mailer /t
```

You hang up and type:

```
send XYZ.WKS to Joe Schmoe now
```

Your SEAdog will ignore all of your normal mail and concentrate on sending XYZ.WKS to Joe while you go out for coffee. It will automatically terminate when it is done. No fuss, no muss, no bother. Elapsed time: a few minutes, at most.

This assumes, of course, that you have a dynamic crash mail event set up.

ADVANCED ROUTING

The **Routing Tags** section describes what the various routing tags mean to SEAdog. You may have noticed that most of the possible routing tags don't mean anything at all. These are available for you to define as you wish. This is done by means of a *routing file*.

A routing file is simply a text file, named **ROUTE.DOG**, which SEAdog checks for any special routing instructions. Routing is described in the routing file by means of a simple, yet powerful, *routing language*.

Here's an example of a routing file:

```
Schedule B
     Send-to 16
     Route-to 16 others

Schedule C
     Send-to all

Schedule D
     Send-to ournet
     Forward-for others
```

The file is free-format text, and can be created and maintained by any standard text editor (such as EDLIN). It contains *routing verbs* which say how the routing is to be done. Most routing verbs are followed by one or more node numbers and/or words which indicate classes of nodes. We'll explain that in a bit. As usual, a semicolon marks the beginning of a comment, and everything from the semicolon to the end of the line is ignored.

Whenever a routing verb needs a list of nodes, you can give one or more nodes separated by spaces or commas. You can put any of the following in a list of nodes:

7	This means node 7 in your own net.
107/7	This means node 7 in net 107.
107/ALL	This means any node in net 107.
ALL	This means any node in any net.
OURNET	This means any node in your own net.
OTHERS	This means any node *not* in your net.
HOSTS	This means node zero in any net.

In addition, you can use the word "except" in a list of nodes, which means that the nodes listed before except for the nodes listed after. For example, consider the following lists:

```
107/all except 107/7
all except 107/all
```

The first list means any node in net 107, except for node 7 in net 107. The second list means any node in any net except for the nodes in net 107. I think you get the point.

Following is a list of all of the routing verbs, and what they mean:

SCHEDULE <tag> [<list>]

This defines the start of the routing commands for a given mail schedule. It is followed by a single character that says what routing tag it is associated with. This is optionally followed by a list of nodes which you are willing to call during this schedule (see *Send-to*).

SEND-TO <list>

This defines a list of nodes which SEAdog is allowed to call during a given schedule.

FORWARD-FOR <list>

This defines what nodes we will forward mail for during this schedule. In other words, we will forward mail from anyone in the <list> to anyone at all.

FORWARD-TO <list>

This defines what nodes we will forward mail to during this schedule. In other words, we will forward mail from anyone at all to anyone in the <list>.

ROUTE-TO <node> <list>

This defines how outgoing mail is to be routed during this schedule. Any messages intended for a node in the <list> will be sent instead to <node>.

NO-ROUTE <list>

This specifies that mail destined for the listed nodes is not to be routed in any way. Use this to turn off the default routing associated with the current routing tag.

PICKUP <list>

This tells SEAdog that it is permitted to pick up mail from any of the listed nodes. See the section on **Low security mail** for more details.

GIVE-TO <list>

This tells SEAdog that it is permitted to give mail to any of the listed nodes when they call. See the **Low security mail** section for more details.

HOLD <list>

This tells SEAdog to hold on to any mail to any of the listed nodes, and to wait for them to pick it up rather than calling them. See the **Low security mail** section for more details.

POLL <list>

This tells SEAdog to call each of the listed nodes and pick up any mail which they may have for you. See the **Low security mail** section for more details.

SEND-ONLY

This defines the mail event as an event during which your node is not expected to receive any calls. The delay time between outgoing calls becomes almost nonexistent. This should only be used if your node is the outbound gateway for a net, as it will be virtually impossible for you to receive mail.

REDIAL <n>

This defines a specific *redial interval*, in seconds. The redial interval is the average number of seconds that SEAdog will wait between placing phone calls. SEAdog will vary from the interval at random by as much as fifty percent either way. In other words, if you specify a redial interval of 60 seconds, then SEAdog will wait from 30 to 90 seconds between calls.

The normal redial interval is two minutes, giving a pause between calls of one to three minutes. *Send only mode* is defined as a redial interval of six seconds (three to nine seconds between calls).

It is possible to set a redial interval of zero seconds, in which case there will be no delay at all between outgoing calls. But we advise against this, as it will then be totally impossible to receive any calls at all. If two systems attempt to call each other, and each system has a redial interval of zero, then neither one will ever reach the other.

NO-REQUESTS

If this statement appears in the routing instructions for any given mail event, then file requests and file update requests received during that event will not be honored.

This is mainly used in the public amateur network to restrict file requests to times other than the normal mail events. Commercial users will probably have no need for it.

EXTERNAL-MAIL <list>

This tells SEAdog that any mail to any of the listed nodes is being sent by some method other than the normal SEAdog mail network. This could be anything at all, such as a commercial data network, or even by mailing a floppy disk. SEAdog will create the mail packet as usual, but will consider it to be sent as soon as it is created. The mail packet will be named "xxxxyyyy.PKT", where "xxxx" is the sum of the originating and destination network numbers in hexadecimal, and "yyyy" is the sum of the originating and destination node numbers in hexadecimal. For example, an external mail packet from node 107/7 to node 100/51 would be named "00CF003A.PKT".

Once the external mail packet is delivered to its destination, by whatever means, it should be renamed to have an extension of ".IN". The receiving SEAdog will then unpack it at the end of its next mail event.

Any mail event that creates external mail packets should be immediately followed by an external event that does something with the packets.

The commands given in the routing file are applied in order. They are added to one another, and to anything already specified by the routing tag.

For example, consider the following:

```
Schedule H
     Send-to 107/8
     Send-to 13/1
```

This tells SEAdog that during schedule H it is allowed to place calls to node 107/8, to node 13/1, and to its own hub.

Any routing commands which precede the first *schedule* verb will apply to *all* mail events.

LOW SECURITY MAIL

SEAdog system security is based on the simple observation that you always know who you are calling, but don't always know who is calling you. For this reason, SEAdog *only* transfers mail from the calling system to the system called.

Well, not always, but that's how it usually works.

If you have mail going, for example, between New York and Los Angeles, then the New York system will call the Los Angeles system to give it the westbound traffic, and the Los Angeles system will call the New York system to give it the eastbound traffic. Hence, two trans-continental phone calls will take place.

If you are more concerned with cost than security, you can bypass the normal security to have all of the traffic go in one phone call. You do this by adding control statements to the route files of the affected systems (see the **Advanced routing** section for more details about creating routing files).

The *pickup* statement tells a SEAdog that it is okay for it to pick up mail when it is making a call. Normally it will only *give* mail when it makes a call.

The *give-to* statement tells SEAdog that it is okay to give a caller his mail. Normally it will only *receive* mail when it answers a call.

Let's take a simple example. Assuming a system with two nodes. Node 1 is New York, and node 2 is Los Angeles. Node 1 has a route file that contains:

```
pickup 2
```

Node 2 has a route file that contains:

```
give-to 1
```

On a given night each node has mail for the other node. When node 1 calls node 2, they will give each other all of the mail going between them.

In the above example, if node 2 called node 1 he wouldn't pick up his mail, and node 1 wouldn't give it to him anyway, because that's what the route files say to do. It would be better if the route file for node 1 contained:

```
pickup 2
give-to 2
```

and the route file for node 2 contained:

```
pickup 1
give-to 1
```

This way, each node knows that it is okay to swap mail no matter which one calls first.

Polling for mail

There are some other things which can be done with the mail pickup mechanism to solve special problems, albeit at the cost of lower security.

The *hold* statement tells SEAdog to hold onto the mail for another node, and wait for him to come get it. The *poll* statement tells SEAdog that it should call another node and try to pick up mail. These work together, as we'll see.

Assume that node 2 has some special problem with phones. He can make calls, but he can't receive calls. Perhaps he can direct dial out, but his incoming calls come through a switchboard. He can still receive mail. Here's how to do it.

Node 1 needs a route file containing the statement:

```
hold 2
```

This tells node 1 that it should not try to call node 2, but instead hold onto his mail and wait for him to come and get it. A "hold" implies giving the mail to a caller, so node 1 doesn't need a separate give-to statement.

Node 2 needs a route file containing the statement:

```
poll 1
```

This tells node 2 that it must call node 1 to see if he has any mail. A "poll" implies picking up mail, so he doesn't need a separate "pickup" statement.

Of course, node 2 could just have a pickup statement, and he'd pick up his mail whenever he called node 1. But SEAdog normally only places a call when it has mail to send. The poll statement tells SEAdog to make a call anyway.

Out-only phone lines

All of this can be used to deal easily with a problem which sometimes arises. It is not uncommon for a business phone line to be able to dial outside numbers directly, but be unable to receive incoming calls without going through a switchboard.

Assume you have twenty nodes in a city, only one of which can receive outside calls. Make that one the hub for the other nineteen. It then holds all mail to the locals, and the locals poll the hub to pick up their mail.

Manual polling

A feature of the MAIL program called *manual polling* can be used to solve several special case problems, again at somewhat reduced security.

Assume you have several nodes which cannot be left running overnight, or which are not at fixed phone numbers (salesmen in the field, perhaps). They can send mail easily enough, but how can they receive any?

Very simple. Set up one machine as a routing hub for these "floating nodes", and leave it running whenever they might be likely to call. At all times its routing is set to hold mail for pickup by these nodes.

A salesman in the field then can hook his laptop computer into a hotel telephone and manually poll the

hub for his mail. Once the call is completed he can peruse the messages at his leasure, and compose replies. He starts up the mailer again to call in and give all of his outgoing mail to his hub. He can now send and receive mail without ever having a fixed location.

Selective holds

Holding mail for pickup can be made selective as well, at least for file transfers. ROBOT and SEND both recognize a "HOLD" keyword, which signals that the file transfer is to be placed on hold until the recipient calls in and picks it up. This is different from the "HOLD" routing verb, which places *all* mail on hold.

Like the general HOLD statement in a routing file, a selective hold implies that the addressee will be calling to get whatever is being held. Thus, a held file will be given to the addressee when he calls. You do not need an explicit "GIVE-TO" statement in your routing file.

EXTENDED ADDRESSING

It is unlikely that you will have any use for extended addressing. It is suggested that you skip this section unless you have some reason to think that it might be of interest to you.

Extended addressing is a technique developed at the FidoNet Technical Standards Committee meeting in November of 1986, and is a method of allowing a FidoNet compatible mail system to address mail to systems which are not on its own network (*foreign networks*).

In brief, an extended address field consists of a character string placed in the text of a message. The first character of the string is an ASCII *Start Of Header* (SOH, hex value 01h), which marks the start of the extended address field, followed by a keyword, followed by the extended address, followed by a hard return (CR, hex value 0Dh). Everything from the SOH to the CR is considered part of the extended address field, and is not normally displayed.

A message containing an extended address is normally addressed tc the node which acts as the gateway to whatever foreign network is being addressed. The node which is acting as a gateway will generally require special software in order to convert messages between the different formats.

There are two main types of extended addressing: *Gateway addressing* and *International addressing*. Each of these will be dealt with in turn. This distinction is only in how they are specified. They are actually both implemented the same way.

Gateway addressing

Gateway addressing is usually used to address mail to a foreign mail system. A gateway address is given as the network address of the gateway node, followed by a signal character, followed by the address on the

foreign system. SEAdog currently understands three types of gateway addresses. They are:

1) UUCP addresses
2) Point addresses
3) Literal addresses

Each of these will be dealt with in turn.

UUCP addressing

A UUCP address consists of the network address of the UUCP gateway node, followed by an exclamation point, followed by the address to use on the UUCP system.

An example of a UUCP address might be:

    132/101!inhp4!encore!vaxine!spark

This would send a message to node 132/101, with instructions to forward the message via UUCP to inhp4!encore!vaxine!spark.

Point addressing

Point addressing is often used to address a node on a *point network*.

Assume you have several private networks and one public network. Each private network has one node who is also on the public network. This node is the gateway node for the private network.

Every node on every network has a node list consisting of the public network plus his own private network, but *not* including any of the other private networks.

The nodes on the private networks are now *points*, and the private networks are now *point networks*. Any node on any network can be addressed by giving the public network address of the gateway node, followed by a decimal point, followed by the *point number*, which is nothing more than the node number on the private net.

For example, the address "107/7.1" means node 1 on the private net whose gateway is listed as node 107/7 in the public network.

All of this sounds complicated, and it is, but it can be handy in certain circumstances. The whole point (pun intentional) is to allow a means of creating a partially decoupled mail system.

SEAdog is normally a fully-coupled mail system. This means that it is not possible to enter a message unless the destination of that message is listed in your own node list. This helps prevent *orphan messages* (messages addressed to nodes that do not exist).

Point nets allow the creation of a "bottom layer" on the mail system that is not coupled. One can enter a message to a point given only that the gateway to the point net exists.

As an example, assume that you are setting up a mail system between five cities. Mail traffic inside of each city will be high, but not much mail will pass between the different cities. In this instance, you might wish to set up each city as a point net. The bulk of the mail traffic will be fully coupled, but every node in every city does not need to keep a full node list of all nodes in all cities.

You will probably have no use for points unless you are setting up a mail system of several thousands of nodes. Also, keep in mind that the nodes which are acting as gateways must run special software in order to properly reroute mail to and from the points.

Literal addressing

Literal addressing is the final catch-all of extended addressing. You can use literal addressing to insert almost any sort of extended addressing you desire.

A literal address consists of the network address of the gateway node, followed by an equals sign, followed by the contents of the extended addressing field. The characters you give for the extended address can contain underlines, which are converted to spaces.

For example, the following two extended addresses are exactly identical:

```
132/101!inhp4!encore!vaxine!spark
132/101=UUCP_inhp4!encore!vaxine!spark
```

This is because a UUCP address field consists of the keyword "UUCP" followed by a space, followed by the UUCP address. The first form is provided only as a convenience.

International addressing

International addressing (more properly called *interzone addressing*) is used to send mail between different FidoNet compatible mail systems, presumably operating in different countries or on different continents. Special software is required to pass mail between different zones.

It is assumed that each system has its own unique *zone number*, and that there exist nodes (called *zone gates*) whose function is to transfer mail between the zones.

A zone address is given as a zone number, followed by a colon, followed by the network address of the destination in the other zone. For example, the address "2:500/1" would indicate node 500/1 in zone 2.

In order for zone addressing to work properly, your own system needs to know what zone it is in. This is specified in the NODE statement in your configuration file. For example:

```
Node 1:107/7
```

would indicate that you are node 107/7 in zone 1.

Interzone mail is always routed to the appropriate zone gate. The zone gates in your own zone are always located in a network or region whose number matches your zone number, while their node numbers are the same as the zone numbers of the zones they gate to. For example, the network address of zone one's gate to zone two would be 1/2. Similarly, in zone two the address of the gate to zone one would be 2/1.

You will probably have no use for zones unless you are setting up a *very* large mail system spanning several continents.

A zone address may also include one other type of extended addressing. For example, a perfectly valid extended address might be:

```
1:132/101!inhp4!encore
```

This would specify the UUCP address inhp4!encore, to be routed through the UUCP gateway located at node 132/101 in zone 1.

SCRIPTING

Scripting is a method for dealing with alternate carriers or non-standard interfaces. The typical user of SEAdog will have no need for scripts, and should probably skip this section.

SEAdog normally sends and receives mail over standard telephone lines. However, in certain circumstances it may be desirable to use an alternate carrier, such as one of the public packet-switched networks. Also, some FidoNet compatible implementations might exist where one must log into a host machine before transferring mail. SEAdog includes a primitive scripting capability to deal with these cases. By "primitive" we mean that it is very basic and simple; it should still be adequate for most needs.

A script file has a name of the form "xxxxyyyy.SCR", where "xxxx" is a four digit hexadecimal network number, and "yyyy" is a four digit hexadecimal node number. For example, a script to be used when calling node 107/7 would be named "006B0007.SCR"

A script is always processed in one of two *modes*:

1) **Sending mode**; In this mode, SEAdog will send any characters found in the script to the remote system, with the exception of the two special characters (see below).

2) **Waiting mode**; In this mode, SEAdog will wait until it receives characters from the remote system to match the characters it finds in the script file, again with the exception of the two special characters (see below).

SEAdog always begins every script in **Sending mode**.

The following characters have special meaning in a script file:

| **Vertical bar**; This causes SEAdog to switch between the two modes.

~ **Tilde**; This causes SEAdog to pause for one second.

In addition, the end of a line in the script is always treated as a single character carriage return.

When SEAdog is in **Waiting mode**, it is *not* waiting for a contiguous string. It is merely waiting, one by one, for a series of characters to appear. In other words, if the script says to wait for "CNCT", and the remote system sends "CONNECT", then that is a match.

Also, case is not important in **Waiting mode**. In other words, if your script says to wait for "CONNECT", and the remote system sends "Connect", then that is a match.

A script will be aborted in either of two cases:

1) Thirty seconds elapse while in **Waiting mode** (text being waited for is not fully received).

2) Carrier is lost (remote system disconnects).

It is expected that a connection with another SEAdog or compatible system has been established when the end of the script is reached.

When using a script, a connection does not count as a "connect with carrier" unless the script is fully processed. If the script is aborted for any reason, then the call is considered to be a "connect without carrier." See the description of the ***Retry*** configuration verb in the **Installation** section for information about connects with and without carrier.

Example

Here is an example of how you would use a script to contact node 107/7 using GTE Telenet's PC Pursuit service:

First, the entry for node 107/7 in your NODELIST.BBS file should contain your local Telenet access phone number instead of the "real" phone number for node 107/7.

Then, you should have a file named 006B0007.SCR (see above) in your SEAdog work area. It's contents should look something like:

```
~
D
|TERMINAL=|D1
|@|c dial201/12,youridcode
|PASSWORD =|yourpassword
|CONNECTED
|~ATZ
|OK|~ATDT 472-8065
|CONNECT
```

Here's what the above script is saying:

1) When connection is established, pause for one second, then send "<cr>D<cr>".

2) Wait for Telenet to ask "TERMINAL=", and reply "D1<cr>".

3) Wait for Telenet's "at sign" prompt, and then give the connect command.

4) Wait for Telenet to ask for your password, and then give it.

5) Wait for the remote modem to be connected. Then pause for one second, and give an "ATZ" command.

6) Wait for the remote modem to report "OK". Then pause for one second, and give the command to dial the destination system.

7) Wait for the remote modem to report connection, and then proceed with a normal SEAdog mail transfer.

TECHNICAL DATA

This section describes various technical aspects of the SEAdog mail system. It is intended to help programmers and other technical people in the design and implementation of utilities and programs to work with SEAdog, and to assist technical support people in general. The average user can skip this section.

SEAdog internal files are binary images of internal data structures. Since SEAdog is written in C, these structures will be defined in terms of the C language. These files are read and written using the fread() and fwrite() functions. Integers are stored in the usual Intel "big-endian" format (least significant byte first). Character strings are stored left to right, ending with a null (usual C convention). All structures are non-aligned. Data sizes are as follows:

Type	Size
char	1 byte
short int	2 bytes
int	2 bytes
long int	4 bytes

Message files

Message files are named "<n>.MSG", where <n> is the number of the message with no leading zeroes. Each file begins with a binary message header, as follows:

```
struct msghdrs                          /* message header structure */
{   char m_from[36];                    /* who from */
    char m_to[36];                      /* to whom */
    char m_subj[72];                    /* subject */
    char m_date[20];                    /* date of message */
    int m_times;                        /* number of times read */
    int m_dnode;                        /* destination node */
    int m_onode;                        /* originating node */
    int m_cost;                         /* message cost, in cents */
    int m_onet;                         /* originating net */
    int m_dnet;                         /* destination net */
    int m_caca[4];                      /* extra space */
    unsigned m_rep;                     /* thread to previous */
    int m_attr;                         /* message attributes */
    int m_up;                           /* thread to next */
}   ;
```

84 BYTES

The header is followed by the text of the message, which is stored as a string of characters ending with a null. The text may or may not contain carriage returns, each of which may or may not be followed by a linefeed. Any of these carriage returns may be "soft". If the high order bit (0x80) of the carriage return is set, then it is a *soft return*. Line feeds and soft returns should be ignored.

Packet files

When SEAdog prepares to send mail, it creates one *mail packet* for each system it will call. Outgoing mail packets are named "<n>.OUT", where <n> is the number of the packet. Inbound packets are named "<n>.IN", where <n> is an integer, starting at one. SEAdog will attempt to unpack all files with an extension of ".IN" at the end of every mail event.

A mail packet consists of a packet header, followed by zero or more packetized messages, followed by an end of packet marker. The packet header is formatted as follows:

```
struct_pkthdrs                          /* packet header structure */
{   int ph_onode;                       /* originating node */
    int ph_dnode;                       /* destination node */
    int ph_yr, ph_mo, ph_dy;            /* date packet was assembled */
    int ph_hr, ph_mn, ph_sc;            /* time packet was assembled */
    int ph_rate;                        /* packet baud rate */
    int ph_ver;                         /* packet version */
    int ph_onet;                        /* originating net */
    int ph_dnet;                        /* destination net */
    int ph_rsvd[17];                    /* reserved for future use */
}   ;
```

This is a "type 2" packet, meaning that the packet version should be set equal to 2. Type 1 packets are not created.

Each message in the packet begins with an abbreviated packetized message header. This header consists of fixed field data, followed by variable field data. The fixed field data is as follows:

```
struct_pktmsgs                          /* packetized message headers */
{   int pm_ver;                         /* message version */
    int pm_onode;                       /* originating node */
    int pm_dnode;                       /* destination node */
    int pm_onet;                        /* originating net */
    int pm_dnet;                        /* destination net */
    int pm_attr;                        /* message attributes */
    int pm_cost;                        /* message cost, in cents */
}   ;
```

This is followed by the variable field data, which is a series of null terminated character strings, as follows:

Field	Maximum length
Date	20 characters
To whom	36 characters
Who from	36 characters
Subject	72 characters
Message text	**

** The maximum length of the message text is in theory unlimited, but in practice it should not exceed a few thousand characters.

This is a type 2 packetized message header, meaning that the header version should be set to 2. Type 1 headers are not created.

The mail packet is terminated with a type zero message header, which is two bytes in length. In other words, the packet ends with two null bytes.

Message attributes

Message headers, both in message files and in mail packets, contain an integer field holding *message attributes*. These are bit values that select various properties of the message. They are defined as follows:

```
#define MSGPRIVATE 0x0001          /* private message */
#define MSGCRASH   0x0002          /* crash priority message */
#define MSGREAD    0x0004          /* read by addressee */
#define MSGSENT    0x0008          /* sent okay */
#define MSGFILE    0x0010          /* file attached */
#define MSGFWD     0x0020          /* being forwarded */
#define MSGORPHAN  0x0040          /* unknown destination */
#define MSGKILL    0x0080          /* kill after mailing */
#define MSGLOCAL   0x0100          /* true if message entered here */
#define MSGHOLD    0x0200          /* true to hold for pickup */
#define MSGX2      0x0400          /* reserved -- sent */
#define MSGFREQ    0x0800          /* requesting a file */
#define MSGRREQ    0x1000          /* return receipt requested */
#define MSGRRCT    0x2000          /* return receipt */
#define MSGAREQ    0x4000          /* request audit trail */
#define MSGUREQ    0x8000          /* requesting a file update */
```

The following attribute bits are included in the packetized message header:

```
MSGPRIVATE
MSGFILE
MSGCRASH
MSGX2
MSGRREQ
MSGRRCT
MSGAREQ
```

All other attribute bits are masked off, and are not sent to other systems.

Any of the following bits, when set, indicate that the message should be deleted once it has been sent:

MSGKILL
MSGFWD
MSGRRCT
MSGFREQ
MSGUREQ

Network and node lists

SEAdog reads the text node list and generates two binary files based on it. These are NODELIST.DOG, which is the list of all valid nodes, and NETLIST.DOG, which is an index into the node list by networks. These files are regenerated whenever SEAdog detects that the text node list is newer than the binary lists, as shown by the DOS date/time stamps.

SEAdog may also create a fast node list index named INDEX.DOG. This file, if it exists, contains data forming an index to the network and node list files. Each entry in the index has the following format:

```
struct nidxs                        /* index record structure */
{   int idxnet;                     /* net number */
    int idxnode;                    /* node number */
    long netiptr;                   /* net list pointer */
    long nodeiptr;                  /* node list pointer */
    long idxspace;                  /* expansion area */
}   ;
```

These records are sorted by ascending order of network number and node number. The index might not include all networks and nodes.

The network list, NETLIST.DOG, includes all known networks, and is used as an index into the node list.

Each record of the network list has the following format:

```
struct netls                        /* netlist entry format */
{   int netnum;                     /* network number */
    char netname[14];               /* name of node */
    char netcity[40];               /* net host location */
    int havehost;                   /* true if net has a host */
    int nethost;                    /* node number of host */
    int havegate;                   /* true if net has a gate */
    int netgate;                    /* node number of gate */
    long nodeptr;                   /* pointer into node list file */
    int numnodes;                   /* number of nodes in net */
}   ;
```

The node list, NODELIST.DOG, includes all known nodes. Each record has the following format:

```
struct nodels                       /* nodelist entry format */
{   int nodenum;                    /* node number */
    char nodename[14];              /* name of node */
    char nodecity[40];              /* location of node */
    char nodephone[40];             /* phone number of node */
    int havehub;                    /* true if node has a hub */
    int nodehub;                    /* node number of hub */
    int nodecost;                   /* cost of call, in cents */
    int nodebaud;                   /* maximum baud rate for node */
}   ;
```

The system file

SEAdog maintains a system data file, named SYSTEM.DOG, in which it stores information it wishes to remember between runs. SEAdog will create a system file whenever it fails to find one present on the disk, or when the one it finds is invalid.

The contents of the system file is specific to any given version of SEAdog, and may change from time to time. Other programs should *not* read the system file, or count on its format or contents.

The system file should *not* be modified by any other program for any reason.

The FidoNet Protocol

As was stated earlier, SEAdog uses the FidoNet Protocol for sending and receiving mail. More detailed information about the FidoNet Protocol is given in the document *A Basic FidoNet Technical Standard*, which is available from the International FidoNet Association at the following address:

International FidoNet Association
PO Box 41143
St. Louis, Missouri 63141
United States of America

Modem defaults

SEAdog knows about certain brands and categories of modems. The MODEM statement in the SEAdog configuration file is used to tell SEAdog what sort of modem it is using. SEAdog recognizes the following modem types:

H12	Hayes 1200 and compatibles
H24	Hayes 2400
COURIER	US Robotics Courier 2400
H12PLUS	Hayes 1200 compatibles with added features

SEAdog assumes certain default values for modem setup, modem reset, and maximum baud rate for each of these modem types. These defaults can be modified by following the initial MODEM statement with additional, more specific MODEM statements. For example:

```
Modem H12
Modem reset AT M0 H1
```

This tells SEAdog to use the default settings for a Hayes 1200 baud modem, but that instead of its normal modem reset sequence it should send the modem the command string "AT M0 H1", which would turn the speaker off and take the phone off hook (thus busying out the line).

The default values for the modem setup string for each type of modem are as follows:

Modem type	Default modem setup string
none stated	AT M0 H0 V1 F1 Q0 X1 S0=1
H12	AT M0 H0 V1 F1 Q0 X1 S0=1
H24	AT &F M0 H0 &C1 &D2 S0=1
COURIER	AT M0 H0 V1 F1 Q0 X5 S0=1
H12PLUS	AT M0 H0 V1 F1 Q0 X2 S0=1

The default values for the modem reset string for each type of modem are as follows:

Modem type	Default modem reset string
none stated	ATZ
H12	ATZ
H24	AT M1 S0=0
COURIER	ATZ
H12PLUS	ATZ

The default maximum baud rate to be used with each type of modem are as follows:

Modem type	Default maximum baud rate
none stated	300 baud
H12	1200 baud
H24	2400 baud
COURIER	2400 baud
H12PLUS	1200 baud

Extended addressing

Extended addressing is a technique for addressing mail to foreign mail systems that was developed during the FidoNet Technical Standards Committee meeting in November of 1986. It allows for almost any sort of address extension that may be desired.

An extended address field is a special field which is contained in the text area of a message. While message text is normally constrained to the limits of printable ASCII text, an extended address field begins with an ASCII Start Of Header (SOH, hex 01h). This is followed by normal text (subject to the usual constraints of message text) and ends at the next hard return (CR, hex 0Dh). Extended addressing fields may be located anywhere within the text of a message.

The first "word" of an extended address field is the field type. The remainder of the field is the data area. SEAdog understands the following field types:

UUCP	Indicates a UUCP address.
TOPT	Indicates a point address.
FMPT	Indicates a point source.
INTL	Indicates an international address.
AREA:	Indicates an EchoMail area label.
SEEN-BY:	Indicates EchoMail tracking data.
Via	Indicates mail routing tracking data.

The data field of a UUCP address consists of the path that the message is to follow over the UUCP mail system. The data field for a TOPT or a FMPT is the number of the node on the private network.

The data field for an INTL field has a very specific format, and consists of two interzone addresses separated by a space. Each of these is given in the full interzone format of "<zone>:<net>/<node>". For example, "1:107/7" would indicate node 107/7 in zone 1. The first address is that of the interzone destination. The second is that of the originator.

Extended address fields are normally not displayed. Hence, almost any sort of data can be "hidden" in an extended addressing field. Specifically, this includes the VIA field, which is used to hide notes on how any given message was actually routed.

The data field of a VIA note is informational, and hence not rigidly defined. Usually, it will take a form on the order of:

Via Node <net>/<node> <time> <date> [<module>]

For example, a VIA note created by the SEAdog mailer would look something like:

Via Node 107/210 04:30 11/24

VIA notes can be especially useful in diagnosing mail routing problems. Usually, the date and time given in a VIA note indicates when the message actually left the forwarding system.

The content and format of the EchoMail data fields (AREA: and SEEN-BY:) are defined by the needs of the EchoMail conferencing system. The SEAdog MAIL program is itself minimally compatible with the New Hampshire conventions for EchoMail conferencing. This can be expanded to full EchoMail functionality by use of multiple message areas and by use of external utilities.

Replacement serial device drivers

The SEAdog mailer comes equipped with a high performance serial device driver for the IBM-PC using an Intel 8250 UART or compatible device. This device driver should be suitable for most compatible systems, though a few MS-DOS machines do use incompatible hardware.

It is still possible to use SEAdog on a system having incompatible serial ports. This is done by installing a replacement serial device driver.

The SEAdog mailer communicates with the serial device driver by issuing function calls through INT 14h. This is the same interrupt used to request serial port services through the IBM BIOS, but the system default serial driver in BIOS is not capable of performing serial I/O for a high-performance application.

To make full use of SEAdog, including the full benefits of the SEAlink sliding window protocol, the serial device driver should be fully interrupt driven, with both input and output buffers. The input buffer should be at least 1024 bytes long, and the output buffer should be at least 256 bytes long. Buffer sizes larger than this can be used, but SEAdog is unlikely to derive any benefit from them.

Funtion calls to the comm driver are all made via interrupt 14h. Registers are used as follows:

AH := The number of the function to be performed. See below.

AL := Data for the function, if applicable.

DX := The number of the serial port to use. Zero for COM1, or one for COM2. See notes below.

Following are the function calls that must be supported by the replacement comm driver for INT 14h:

AH := 0 Set baud rate

This works the same as the equivalent BIOS call, except that it ONLY selects a baud rate. Other parameters are always set to eight data bits, one stop bit,

and no parity. You can support other modes if you wish, but SEAdog won't be making use of them.

AH := 1 Transmit character

AL contains the character to be sent. On return, AX is set as in a status request.

AH := 2 Receive a character

Does not return until a character is received. Returns with the received character in AL, zero in AH.

AH := 3 Request status

Returns with the line and modem status in AX. Status bits are the same as the equivalent BIOS call.

NOTE: Most of these status bits can be taken directly from the 8250 status ports. However, two in particular should be handled to reflect buffer states if serial I/O is buffered. These are:

2000h - Transmit Holding Register Empty
0100h - Data Ready

The Transmit Holding Register Empty flag should be set or reset to indicate whether or not the serial driver can accept more data. In other words, it should reflect the status of the transmit buffer (if any).

The Data Ready flag should be set or reset to indicate whether or not the receive buffer contains any data.

These two flags should *not* reflect the state of the hardware, but of the serial driver itself. The sense of these flags are:

THRE says, "Yes, I can accept data now."
DR says, "Yes, I have data for you now."

AH := 4 Initialize driver

This is used to tell the driver to begin operations, and to check that the driver is installed. This function will be called before any other actions are performed. On entry, DX will be set with the number of the serial port to use (0=COM1, 1=COM2) on all future function calls.

AH := 5 Deinitialize driver

This is used to tell the driver that serial port operations are ended. This function will be called when no more functions will be used.

AH := 6 Raise/lower DTR

This function is used to control the DTR line to the modem. AL=0 means lower DTR (disable the modem), and AL=1 means raise DTR (enable the modem). No other function should alter DTR.

AH := 7 Return timer tick parameters

This is used to determine the parameters of the timer tick on any given machine. Three numbers are returned:

AL = Timer tick interrupt number
AH = Ticks per second
DX = Milliseconds per tick

The appropriate values for an IBM-PC would be:

AL = 8
AH = 18
DX = 55

AH := 8 Flush output

This is used to force any pending output. It should not return until all pending output has been sent. This does not apply to drivers which do not buffer output.

AH := 9 Purge output

This is used to purge any pending output. Any output data which has not yet been sent should be discarded. This does not apply to drivers which do not buffer output.

AH := 10 Purge input

This is used to purge any pending input. Any input data which has not been read yet should be discarded. This does not apply to drivers which do not buffer input.

AH := 11 Transmit character without waiting

This is similar to function 1, *Transmit character*, except that it always returns immediately, even if the character cannot be accepted right away. On return, AX should contain a one if the character could be accepted, or a zero if it could not.

AH := 12 Scan for character

This is similar to function 2, *Receive a character*, except that it is non-destructive. If no data is available, then it should return with 0FFFFh in register AX. Otherwise, it should return with the character value in AL and zero in AH, and the character should still be available for reading.

AH := 13 Scan keyboard

This is used to detect whether or not a key has been pressed at the keyboard. If no keystroke is available, then it should return with 0FFFFh in AX. Otherwise, it should return with the value of the keystroke in AX (scan code in AH, ASCII value in AL), and the keystroke should still be available for reading.

AH := 14 Read keyboard

This should read the next keystroke, and return with its value in AX (scan code in AH, ASCII value in AL). If no keystroke is waiting, then it should wait until a key is pressed.

AH := 15 Flow control

This is used to set and reset different forms of flow control. The different types of flow control are defined by the following bits in AL:

01h XON/XOFF switch
02h CTS/RTS switch
10h XON/XOFF mask
20h CTS/RTS mask

On entry, The mask bits will be set to indicate what types of flow control are to be affected by this call. The switch bits will be set to indicate which of those methods should be activated.

In other words, if the calling program wishes XON/XOFF flow control to be activated, and CTS/RTS flow control to be deactivated, it would call this function with the value 31h in AL.

On return, AL should be set to indicate what methods of flow control are now active.

Device driver detection:

The presence of a SEAdog compatible serial device driver can be detected by examining the code which handles interrupt 14h. The serial driver should have a one word "marker" six bytes after the address pointed to by the interrupt 14h vector. This is a two byte integer containing the marker value 1954h. This is followed immediately by a one byte value which is the highest function supported by this driver.

This structure is very easy to implement in assembler, as follows:

```
entry:  jmp  start
        org  entry+6
        dw   1954h
        org  entry+8
        db   14

start:  <etc>
```

If a serial device driver does *not* contain the marker value and the following support level byte, then SEAdog will use its own default device driver. The SEAdog mailer requires support up to and including function 14.

Implementation notes:

Function calls 4 and 5 are used to turn the serial driver on and off. SEAdog will not do anything to the serial driver until it has given it a function 4 call to initialize the driver.

SEAdog will call function 5 to turn off the driver when it exits and when it invokes another program. It will then call function 4 again to restart the driver when it regains control.

The port number will be available in register DX on every call, but SEAdog will not change ports without first calling function 5 for the old port, and then function 4 on the new port. Hence, the port number may be ignored on any function call other than function 4. In fact, it may be ignored completely if you only have one serial port.

This type of serial interface driver is known as a FOSSIL driver. If a FOSSIL driver already exists for your machine, then you should be able to use it with no problems.

USING SEADOG WITH TBBS

If you've never heard of TBBS, then don't bother reading this section.

It is possible to use SEAdog as a front end for TBBS versions 2.0 and later. TBBS is loaded first, and then loads the SEAdog mailer. SEAdog will answer the phone and determine whether or not a caller is trying to send mail, and exits back to TBBS if not.

First, SEAdog must be told to terminate when it gets a non-mail caller. This is done with a BBS statement in the CONFIG.DOG file, like so:

```
bbs *x
```

SEAdog will terminate with an error level of the caller's baud rate divided by one hundred (3 for 300 baud, 12 for 1200 baud, and so on). TBBS knows what to do with this, and will act accordingly.

In addition, you can gain local access to TBBS while SEAdog is running by pressing the "F1" key. This tells SEAdog to terminate with an error level of ten, which TBBS understands as indicating that you want to get on locally.

You should use CEDIT to define external events in TBBS to match your mail events in SEAdog so that TBBS will know when to force callers off to do mail. You will also need to define external events to run PREMAIL before your mail events, and to run POSTMAIL after your mail events. Please refer to the TBBS manual for more details about PREMAIL and POSTMAIL.

You will need to define a message board in TBBS with the name "NET MAIL". This is a special name which lets TBBS know that it should ask for a network address whenever a message is entered in this area. Network mail to and from TBBS will take place only on this message board.

You will need to modify your SET TBBSPATH= environment string to include the directory where the SEAdog system files are kept so that TBBS will be able to find the files it needs. Please refer to the DOS manual for more information about the SET command, and

to the TBBS manual for more information about the TBBSPATH environment string.

You must place a copy of MAILER.EXE in the same directory as TBBS.COM so that TBBS will be able to load it when it runs. Also, TBBS should be invoked with a statement of the form:

```
TBBS /m
```

This tells TBBS to load the SEAdog mailer whenever it is waiting for a caller. If you do not invoke TBBS with the "/m" switch, then it will not load MAILER, and you will be unable to receive mail outside of scheduled mail events (you will not receive crash mail).

TBBS is available from:

eSoft, Inc.
4100 S. Parker Rd. #305
Aurora, CO 80014
United States of America

Please refer to your TBBS manual for more information about how to configure TBBS to work with SEAdog.

SEAdog's default banner states that it is a private mail system, and asks the caller to hang up. You will want to change this. You can use either or both of two methods. The first is the BANNER statement in your configuration file. Whatever follows the word "BANNER", up to the end of the line or a semicolon, is displayed instead of the default banner. A typical example might be:

```
BANNER SEAboard system -- stand by for TBBS
```

You can also create a text file named BANNER.DOG, which will be displayed instead of the banner string to any callers outside of mail events. This file may contain anything you like, and may be as long as you like, but we recommend that you keep it short, as SEAdog banners cannot be interrupted.

USING SEADOG WITH FIDO

If you've never heard of Fido, then don't bother reading this section.

It is possible to run Fido under SEAdog. SEAdog will answer the phone and determine whether or not a caller is trying to send mail, and pass them on to Fido if not.

First, SEAdog must be told to pass non-mail callers on to Fido. This is done with a BBS statement in the CONFIG.DOG file, like so:

```
bbs RUNBBS *b
```

SEAdog will pass a human caller to Fido by invoking another generation of DOS and giving it whatever command you specify in your BBS statement.

Fido must now be told that it is being passed a user, and that it should return control to SEAdog when the user is finished. This is done by way of the /N and /E switches. Your RUNBBS.BAT should contain a statement something like this:

```
fido_ibm %1/n 5/e
```

The "%1/n" (where %1 gets replaced with a baud rate) tells Fido that it already has a user at the specified baud rate. The "5/e" tells Fido that it should terminate with an error level of 5 when it is finished with the user. Fido won't terminate if you use "0/e".

You will probably want to add other switches as needed to configure your system. See the Fido manual for more details.

You will need to define external events in Fido to correspond to all of your SEAdog events. Otherwise, Fido won't know when it has to force a user off. For example, if you are using the events given above, then you should tell Fido that it has an external event starting at 0300, with a 180 minute window. This event should exit with the same error level as that of the "/e" switch, since they both mean the same thing (return to SEAdog). We suggest that you have SEAdog handle all of your "normal" external events. Don't

forget to reschedule Fido's external events when changing to or from Daylight Savings Time.

If you wish to use Fido's O)utside or sysop zero commands, then you'll need to set your error levels and your batch file properly. Here's an example of a RUNBBS.BAT for Fido with the O)utside and sysop zero commands:

```
echo off
:loop
fido_ibm %1/n 5/e 10/w 15/a
     if errorlevel 15 goto outside
     if errorlevel 10 goto dropdos
     goto seadog

:outside
watchdog on
remsysop /c
watchdog off
goto loop

:dropdos
watchdog on
ctty com1
echo Type EXIT to return to Fido.
command
ctty con
watchdog off
goto loop

:seadog
```

You can't just "run off the end" for remote system access, because DOS will exit and return to SEAdog. Instead, you should invoke a new generation of DOS b giving the "COMMAND" command. It will terminate whe you type "EXIT", and you will loop back into Fido. The meaning of "%1" will be retained, and Fido will start up at the proper baud rate.

If you do *not* need any DOS access from Fido, then yo don't need a RUNBBS.BAT file at all. Instead, you c have SEAdog invoke Fido directly. The BBS statement in your CONFIG.DOG file would then look something li this:

```
BBS fido_ibm *b/n 5/e
```

SEAdog's default banner states that it is a private mail system, and asks the caller to hang up. You will want to change this. You can use either or both of two methods. The first is the BANNER statement in your configuration file. Whatever follows the word "BANNER", up to the end of the line or a semicolon, is displayed instead of the default banner. A typical example might be:

```
BANNER SEAboard system -- stand by for Fido
```

You can also create a text file named BANNER.DOG, which will be displayed instead of the banner string to any callers outside of mail events. This file may contain anything you like, and may be as long as you like, but we recommend that you keep it short, as SEAdog banners cannot be interrupted.

If Fido gets "stuck", which can easily happen if a caller hangs up without being logged on and validated, then don't worry. Fido will return to SEAdog properly after the next caller or at the next event. You will not, however, be able to receive mail while Fido is in control.

Fido is available from:

Fido Software
Tom Jennings
2269 Market St. #118
San Francisco, CA 94114

THE PUBLIC AMATEUR NETWORK

There exists a public amateur network of electronic mail hobbyists who send and receive mail using the FidoNet Protocol. Indeed, the FidoNet Protocol derives its name from this network, which is usually called *FidoNet*, since it originally began as a network of Fido bulletin boards. There are now many mail systems that can (and do) participate in this network, including SEAdog.

If you have purchased SEAdog with an eye to participating in this network, then there are several things you should do. First and foremost, you should be aware of the various policies established to govern the actions and interrelations of nodes on the network. These are described in the *FidoNet Policy and Procedures Guide*, which is available from the International FidoNet Association (IFNA). This document describes such things as how to obtain a node number, how to obtain node list updates, how to resolve disputes with other nodes, and a host of other matters that anyone participating in the public amateur network should be aware of. This document is available from:

International FidoNet Association
PO Box 41143
St. Louis, Missouri 63141
United States of America

The International FidoNet Association is a non-profit corporation which performs many useful services, including publishing a weekly electronic newsletter and keeping an eye on possible legislation which may affect users of electronic mail. We urge anyone interested in using electronic mail or in operating an electronic mail network to support IFNA, even if you do not join the public amateur network.

You should use somewhat different mail events when operating on FidoNet than are appropriate in a commercial mail network. We recommend some variation of the following:

```
Event base   4:00              ;Based on start of National Mail Hour
Event H all -1:00 -0:30        ;Locals to hubs
Event G all -0:30  0:00        ;Hubs to outbound gates
Event T all  0:00  1:00        ;National Mail Hour
Event V all  1:00  1:30        ;Inbound gates to hubs
Event W all  1:30  2:00        ;Hubs to locals
```

Adjust the event base as needed for your time zone and to allow for Daylight Savings Time. Remember: if you advance your system clock, you should also advance your event base, and vice versa. FidoNet does *not* observe Daylight Savings Time.

Check with your network host for the exact times on the events surrounding National Mail Hour. If your net does not have hubs, then you can leave out events H and V. If you do not have an outbound host, then you can leave out events H and G. If you are in a region, then we recommend a single event (with the tag "A") covering National Mail Hour. Please refer to the **Routing tags** section for more details about the meanings of the various tags.

You will need to translate the St. Louis format node list into the proper format for SEAdog. We suggest you use the XLATLIST utility for this purpose.

If you have an outbound host, then you will need to modify the St. Louis node list. (FidoNet inbound and outbound hosts are sensibly equivalent to SEAdog inbound and outbound gateways.) You would do this by placing an OGATE statement in your XLATLIST control file. For example, if your outbound host is node 107/16, then your XLATLIST.CTL file should contain:

```
OGATE 107/16
```

You should place XLATLIST and ROUTEGEN in SEAdog mode by placing a SEADOG statement in the XLATLIST control file. We advise that you use SEAdog implicit routing wherever possible. Detailed routing files can be used, but are not needed.

CHANGES IN VERSION 4.0

Version 4.0 of SEAdog contains substantial changes from earlier versions. The major changes will be described in this section.

New features

The biggest new feature, of course, is the use of the SEAlink protocol for mail and file transfers. SEAlink is a sliding window variation of XMODEM that retains full backwards compatibility with earlier versions, as well as with other FidoNet compatible mail systems.

SEAlink was developed expressly for use in version 4.0 of the SEAdog electronic mail system, but has also been made available to developers of other communications software. Even while still being tested, SEAlink produced such dramatic improvements in system throughput that it was adopted as an extended protocol standard for use in FidoNet compatible mail systems. SEAlink is up to 20% faster on local connections, and up to 100% faster on long distance connections which involve satellites (such as overseas calls).

Other new features include:

- Improved modem support and modem error recovery.
- Improved support of multiple message areas.
- Multiple levels of log file detail.
- Improved printer support.
- Support of FidoNet extended addressing fields, with the ability to show or hide the extended address fields.

- Support for 19.2 kilobaud modems.
- Improved support for pulse dialing.
- New utility functions, TELL and ULMAINT.
- Improved comm driver support for high speed machines.
- Improved support of FidoNet compatible polling.
- Modem setup and reset strings can now be up to 99 characters long.
- Script capability, to allow the use of alternate carriers.
- Function keys now cause MAILER to exit with an error level.
- The Alt Q and Alt R commands were added to MAILER to allow more control over mail events.
- Pressing the question mark key when MAILER is performing a mail event now causes a status report to be displayed.
- Crash external events were added.
- The BBS event modifier was added, allowing simultaneous mail and BBS access.
- The ">" form of message reply was added to MAIL.
- Automatic aliasing was added to MAIL.
- A self-addressing mode was added to MAIL, allowing mail to be addressed to your own node.
- The purge function of MAIL was enhanced to allow automatic purging of orphaned messages.

Incompatible changes

Some changes were made in version 4.0 which are not compatible with earlier versions of SEAdog. These are all related to the configuration file (CONFIG.DOG), as follows:

- The BAUD, COM, MODINIT, and MODFINI statements have been eliminated, and are now part of the MODEM statement.
- The PRNINIT and PRNFINI statements have been eliminated, and replaced by the PRINTER statement.
- The HELP statement no longer takes a default help level. It is now used *only* to specify the directory in which the help files are located.

INDEX

BLANK PAGE

DISCLAIMER

NO WARRANTIES ARE EXPRESSED OR IMPLIED BY SYSTEM ENHANCEMENT ASSOCIATES, INC., PARTICULARLY WARRANTIES OF MERCHANTABILITY OR OF SUITABILITY FOR ANY PARTICULAR PURPOSE, WITH REGARD TO THE MATERIAL HEREIN.

System Enhancement Associates, Inc. assumes no liability for any errors, inaccuracies, or omissions found herein or in the accompanying software, or for any damages resulting from the use of this material.

System Enhancement Associates, Inc.
21 New Street, Wayne NJ 07470